Beköszöntő

Jelentjük, itt a tavasz!

Ami áprilisi számunk tartalmát illeti, egyetlen fő téma taglalása helyett igyekeztünk inkább körképet adni a ***Linux*** alkalmazási területeiről.

Szó esik szuperszámítógépekről, digitális hangstúdióról, számítógéppel megvalósított mozgásérzékelőről, programok optimalizálásáról, egy grafikai keretrendszerről, és egy régi jó adatbázis kezelő új változatáról.

Folytatódik az előző számban indult, a jogosultságok központi kezeléséről szóló sorozat is.

Magyar szerzőink is kitettek magukért. ***Dudás Ágnes*** most a kész programok visszafejtésének „jogkövetkezményeit" tárgyalja, ***Garzó András*** pedig folytatta az útválasztók működési logikájának tárgyalását. ***Illés Viktor Sambáról*** szóló sorozata immár a harmadik részéhez érkezett, ***Auth Gábor*** pedig a 6. résznél tart a ***FreeBSD*** bemutatásával.

Van szerencsénk bemutatni két új szerzőnket is. Mindketten középiskolai tanárok, és mindketten a ***Linux*** iskolai felhasználásáról írtak.

Szabó Zoltán pingvint klónoz, vagyis bemutatja, hogyan telepíthetjük ezt az operációs rendszert egy egész géptereben, viszonylag gyorsan és fájdalommentesen.

Jászberényi József a levélszemét elleni küzdelemre fejlesztett ki egy hatékony módszert. Cikkéből megtudhatjuk, milyen védekezési lehetőségeink vannak, ha a levélkiszolgálónkhoz ezerszám érkeznek a hibás címzéssel ellátott levelek, és ettől a teljes levelezés megbénul. Előrebocsátom: a megoldás meglepően egyszerű.

Fábián Zoltánnak ebben a számban rögtön két cikke is megjelenik. Az egyik a korábbi GIMP-es sorozat befejezése, a másik pedig egy olyan könyvtárral foglalkozik, amellyel könnyen és gyorsan építhetünk fel látványos grafikus felhasználói felületeket.

Kellemes időtöltést kíván
a Linuxvilág stábja!

Mandrake – Conectiva házasság

A *Mandrakesoft S.A.* megerősítette, hogy 2,3 millió dolláros részvényügylettel felvásárolta Dél-Amerika

vezető linuxos vállalatát, a brazíliai *Conectivát*. A felvásárlás révén a *Mandrake* Európából Dél-Amerika felé tud majd terjeszkedni, valamint bővítheti kutatói bázisát is.
A *Conectiva* 1995-ben alapult, jelenleg három irodában összesen 60 főt foglalkoztat, pénzügyi helyzete stabil, elmúlt évi bevétele 2,2 millió dollár volt.
➲ www.conectiva.com.br

PostgreSQL 8.0

A *PostgreSQL Global Development Group* bejelentette „a világ legfejlettebb nyílt forrású, objektumorientált-relációs adatbázis-kezelő rendszerének" 8.0-s változatát. A *PostgreSQL* méretezhetőségben, a szolgáltatások választékában és teljesítmény tekintetében egyaránt felülmúlja elődeit. Újdonságai a komoly kereskedelmi adatbázis-kezelő megoldásokat idézik: a mentési pontok támogatása, ezek segítségével a tranzakciókat megadott pontra lehet visszaállítani, mégsem kell teljes egészében megszakítani őket; az adott időpontra való visszaállítás lehetősége, amelyet az adatbázis-kezelő a tranzakciós naplók alapján végez el; a táblaterek, amelyek lehetővé teszik a nagyméretű, akár sok GB-os táblák külön meghajtóra helyezését, és ezzel hozzájárulnak a műveletek felgyorsításához; valamint a natív windowsos változat, mely a *Windowshoz* kötődő felhasználók számára a továbbiakban szükségtelenné teszi az emulációs réteg használatát, és értelemszerűen nagyságrendi teljesítménynövekedést hoz. A *PostgreSQL* mellett döntőknek a kiegészítők tájékán is érdemes körülnézniük, ugyanis ezek köre is számottevően bővült az előző évhez képest, valamint a tárolt eljárások megvalósítására használható programnyelvek támogatásán is sokat javítottak.
➲ www.postgresql.org

Harmadik generációra fel!

A *Samsung* bejelentette, hogy több más gyártó közreműködésével elkészítette első referenciarendszerét a harmadik generációs (UMTS) hálózatokhoz tervezett, linuxos mobiltelefonokhoz. Az új alaprendszer támogatja a következő generációs hálózatok szolgáltatásait, mint a videohívás és a videofolyamok továbbítása, továbbá Java alapú alkalmazások futtatására is képes.

A Cannes-ban rendezett *3GSM World Congress* kiállításon a Samsung kész termékeket is bemutatott, a szintén 3G hálózatokra csatlakozó, 90x44x25 mm-es, 1 MP felbontású kamerával, belső memóriával és *Bluetooth*-csatolóval ellátott *SGH-Z500* jelenleg a világ legkisebb 3G mobiltelefonja; az *SGH-Z300* két hangszóróján keresztül 3D hangot adó zenelejátszóval rendelkezik; míg az *SGH-Z130* egyedülálló, forgatható, szélesvásznú kijelzőt kapott.

Szalonképes PHP

Az *IBM* szövetségre lépett a *PHP* nyelvéről ismert *Zend Technologies Ltd.*-vel – céljuk a dinamikus, személyre szabott tartalmat szolgáltató weboldalak fejlesztésének megkönnyítése.
Az *IBM* egy *Zend Core for IBM* nevű, a *DeveloperWorks* oldalról a második negyedévtől kezdődően szabadon letölthető termék formájában fogja kínálni a *PHP*-t és saját, *Cloudscape* nevű adatbázisrendszerének nyílt forrású változatát. Az *IBM* lépése fontos lehet a *PHP* szempontjából, hiszen így a nagyvállalati piaci szegmensben is elismert megoldássá válhat.

AMD vagy Intel?

Az *AMD* egy pillanatra sem szeretné elveszíteni lendületét, központjában nemrég nyilvánosan bemutatta első kétmagos *Athlon 64* processzorát. A cég többmagos, 90 nm-es eljárással készülő *AMD64* lapkáit a kiszolgálóktól kezdve a munkaállomásokon keresztül egészen az ügyfélrendszerekig számos területen szeretné elérhetővé tenni. Hangsúlyos elem, hogy a végfelhasználók számára biztosítani kell a többmagos rendszerekre való átállás zökkenőmentességét, ennek egyik jele, hogy az újabb lapkák a jelenlegi alaplapi foglalatokban és áramellátó egységekkel is használhatók lesznek.
Természetesen szó sincs arról, hogy az egymagos processzorok egy csapásra eltűnnének a kínálatból, az *AMD* elképzelései szerint az *Athlon 64* és az *Athlon 64 FX* lapkák még jó ideig jelen lesznek, elsősorban az otthoni felhasználók gépeiben.
A nagy rivális, az *Intel* is tesz arról, hogy szerepelhessen a hírekben. Már itthon is kaphatók az új, 600-as sorozatba tartozó *Pentium 4* processzorok, amelyek 1 MB gyorsítótárral ellátott elődjeikhez képest immár 2 MB gyorsítótárral rendelkeznek, továbbá támogatják az *NX* jelző használatát és a 64 bites memóriakezelő kiterjesztéseket is.
A *Prescott* magos *Pentiumok* lassan félelmetessé váló energiafogyasztását az új maggal szerelt példányokban az órajelet a terhelés függvényében változtató *SpeedStep* megoldás igyekszik féken tartani. A 2 MB gyorsítótárral ellátott processzorok a tesztekben meglepő módon számottevő teljesítménybeli előnyt nem tudtak felmutatni, ellenben áruk 20-30 százalékkal meghaladja az elődökét.
Nem árt megjegyezni a *Pentium D* nevet sem, így fogják hívni a jövő negyedévben megjelenő kétmagos *Pentium* processzorokat. A meglévő 775-ös foglalatba illeszkedő lapkák sajátos módon nem fogják támogatni a hiperszálas futtatást, vagyis továbbra is csak két szál futtatására lesznek alkalmasak – vélhetően az *Intel* lassanként akarja csepegtetni a teljesítményt a vásárlóknak. A magonként kettő, összesen négy szál futtatása egyelőre a méregdrága *Extreme Edition* privilégiuma marad.

Linuxos kommunikátor-klón

A német *ROAD* cég a *Nokia* kommunikátoraira emlékeztető linuxos mobiltelefont mutatott be. Az *S101* jelzésű telefon kívülről egyszerű mobiltelefonnal látszik, széthajtva azonban egy teljes értékű *QWERTY* billentyűzet és egy a lehetőségekhez képest nagyméretű kijelző tárul elénk. A telefon 400 MHz-es *Intel XScale* processzort, 64 MB RAM-ot és 64 MB *Flash* memóriát tartalmaz, és a *Qtopia* felületet és személyi adatkezelő készletet futtatja. Súlya 210 gramm, belső kijelzője színes, 640 x 240 képpont felbontású. Kommunikációs szempontból is rendkívül sokoldalú, négysávú (850, 900, 1800 és 1900 MHz), *EDGE*-képes *GSM*-telefon, továbbá képes a *WLAN*, a *Bluetooth* és az infravörös alapú összeköttetések kezelésére.

➲ www.road-gmbh.de

Aki nem tud lemondani róla

A *Win4Lin, Inc.* bejelentette új vezető terméke, a *Win4Lin Pro* szállítását. A *Win4Lin Pro* – a *Linux* rendszermag módosítása nélkül is – teljes értékű futtatási lehetőséget kínál *Linux* alatt a *Windows 2000* operációs rendszerhez, valamint az alatta üzemelő alkalmazásokhoz, illetve – egyelőre kezdetlegesen – támogatja a *Windows XP*-t is. Mivel a *Win4Lin* virtuális számítási környezete a *Windows* egy teljes, változtatások nélküli példányát futtatja, esetében, ellentétben a másik ismert megoldással, a *WINE*-vel, a *Microsoft* által kibocsátott frissítések telepítése sem okozhat gondot. A *Win4Lin Pro* bevezető ára március végéig 99,99 USD – ennyi pénzért vásárolhatunk egy *OEM Windows XP Home Editiont*, vagy *félig* hozzájuthatunk a jóval sokoldalúbb *VMWare Workstation* egy példányához.

Win4Lin

➲ www.win4lin.com

Sun újdonságok

A *Sun Microsystems* bemutatta várhatóan az év közepén megjelenő *StarOffice 8* irodai csomagjának előzetes változatát, valamint a második negyedévre tervezett *Java Desktop System* harmadik kiadását. A *Linux* rendszermag 2.6-os változatára épülő operációs rendszer fejlesztése során nagy hangsúlyt kapott az eszköztámogatás bővítése, a *Microsoft Exchange* kiszolgálókkal való együttműködés és a microsoftos fájlformátumok kezelésének javítása.
A *StarOffice* fejlesztése során szintén a *Microsoft Office* formátumainak jobb kezelése, valamint az általános használhatóság javítására fordították a legnagyobb figyelmet. A *Sun* az importálási és exportálási szűrők és az adatbázis-kezelés javításával, valamint a *Visual Basicben* készült makrók akár automatikus átalakításának támogatásával segíti az *StarOffice*-ra való áttérést, illetve az áttérés költségének és kockázatának csökkentését.
Ugyancsak a *Sun* újdonsága lesz a még idén várható *Trusted Solaris 10* operációs rendszer. A katonai, kormányzati hálózatokba szánt operációs rendszer egyik érdekessége az automatikus, futás idejű ellenőrzés lesz, melynek feladata az operációs rendszer kódjának futtatás közben végzett ellenőrzése. Segítségével, digitális aláírások alkalmazásával megelőzhetők a jogosulatlan módosítások. Hasonló szolgáltatás az elsősorban a kémprogramok elleni védekezést szolgáló *Solaris Secure Execution*, mely képes ellenőrizni, hogy a gyártó általi kiadás óta módosították-e az adott programkódot.
A további finomságok sorában a továbbfejlesztett folyamat- és felhasználókezelés, a prediktív öngyógyítás, a *Solaris Containers*, egy új, a *PKCS#11* szabványra épülő titkosítási keretrendszer, valamint a digitálisan aláírt futtatható fájlok, könyvtárak és rendszermagmodulok. Szintén büszkeségre adhat okot, hogy a *Solaris* előző, 9-es kiadása 15 hónap tesztelés után megkapta a *Common Criteria* minősítést; a 10-es kiadás vizsgálata pedig folyamatban van.

➲ www.staroffice.com
➲ www.sun.com/software/star/staroffice/beta/

Második fokozat

Az *LSI Logic* az első nagyvállalati szintű *SATA-II RAID*-vezérlőt mutatta be a *LinuxWorld* rendezvényen. A *MegaRAID SATA 300 8x* jelzésű vezérlő – a *SATA* szabvány második változatának előírásaiból fakadóan – elődjeihez képest kétszeres, körülbelül 3 Gbps sebességet biztosít, miközben természetesen a *SATA-I*-es merevlemezeket is képes kezelni.
A minden fontosabb operációs rendszert támogató vezérlő érdekessége a kaputöbbszöröző, amellyel elméletileg akár 120 meghajtó kezelésére is képes lehet, igaz, a teljesítmény romlása mellett; továbbá az a kiegészítő akkumulátor, amellyel áramkimaradás esetén akár 72 órán keresztül is képes megőrizni az adatokat. A vezérlő ára várhatóan 550 dollár körül fog alakulni.

➲ www.lsilogic.com

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

LME – A szakma legismertebb egyesülete

Mindenki Ismeri. Mindenki ismeri?

Hiába találja ki az ember, hogy egy-két hét alatt megváltja a világot, a világ csak nem akar olyan gyorsan változni. Lassan frázisokban mondogatjuk, hogy a szabad és nyílt érdekcsoportok immár évtizedek óta küzdenek, hogy átalakítsák a becsontosodott régi rendszert, ami csak nem akar könnyedén eltűnni. Először csak néhány lelkes egyetemi szakember... Ezerszer végigrágtuk már a sztorit.
Magyarországon „hivatalos szinten” az elsők között jött létre a Linux-felhasználók Magyarországi Egyesülete, akkor még egy teljesen más világban.
Az egyesület igazi szakmai baráti csoportosulásként indult, a Nyugati Pályaudvar egyik hírhedt gyorsétkezdéjében, a kilencvenes évek végén. Linux-hívő szakemberek egy kis csoportja alakította, kimondottan a Linux ismertségének, elfogadottságának növeléséért.
A tagok tehát informatikusok voltak, akik szabadidejükben szervezték az egyesületet. Évekig tartó herce-hurca után alakult meg hivatalosan is az egyesület, Péter László elnökségével, de 1998-ban már konferenciát szervezett. Az „öregek” között találjuk többek között Czakó Krisztiánt, Magosányi Árpádot, Milus Jánost, Regős Szabolcsot, Sári Gábort, Szalay Attilát, Varga S. Csabát.
A lelkes világ addig tartott, amíg áttörésként egy tízmillió forintos támogatást nem kapott az egyesület. Ehhez kapcsolódóan ugyanis három munkát kellett letegyen az egyesület az asztalra: Egy magyar nyelvű felülettel rendelkező irodai programcsomagot, egy fordításegítő rendszert, valamint egy könyvsorozatot. E három feladatot sajnos az egyesület csak hiányosan és nehézkesen tudta teljesíteni, ami komoly belső feszültségeket szült.
A vihar eredményeként nyilvánvalóvá vált, hogy az egyesület szerkezete és a célok elérése nincsenek összhangban. Több elnökváltás után a jelenlegi elnökség a piac szerkezetét és az egyéb szervezetek tevékenységét is figyelembe véve igyekszik irányítani az egyesületet. Az érdeklődők számára következzen az egyesület céljainak megfogalmazása, valamint az elmúlt év eredményének rövid összefoglalója.

Szy György

A Linux-felhasználók Magyarországi Egyesülete (LME), mint közhasznú egyesület kiemelt célja

- a Linux operációs rendszer, és az ahhoz kapcsolódó programok (elsősorban a Szabad szoftverek) felhasználásával, alkalmazásával kapcsolatos ismeretek széles körben történő terjesztése, oktatása
- a hazai kutatások, fejlesztések támogatása
- hazai, külföldi és nemzetközi szervezetekkel való együttműködés
- a Linux-felhasználók érdekeinek képviselete (jelenleg is folyamatosan egyeztünk annak érdekében, hogy a többség számára elfogadható szoftver szabadalmak szülessenek)

Közhasznú szolgáltatásainkból tagjainkon kívül más is részesülhet, (akár Ön is!), így tehettük meg hogy jelentős összeggel támogattuk az OpenOffice.org honosítását, az FSN.hu több száz GB-tal történő diszkbővítését (hozzájárulva a legnagyobb közép-európai szabad szoftver gyűjtemény bővítéséhez).
A idén tovább kívánjuk erősíteni tevékenységünket a fenti területeken, különösen az érdekvédelem és az oktatás terén.
A munka oroszlánrészét önkéntesek végzik, működési kiadásaink így is jelentősek. Ha céljainkkal egyetért és megteheti kérjük támogassa adója 1 %-ával egyesületünket!
Támogatását előre is köszönjük!
Az LME 2004. évben, az 1%-os felajánlásokból befolyt összeget a következő célokra használta fel:

- A Fix.tv Linuxportál című műsorának támogatására,
- A számítógéppel megvalósított találmányokról szóló direktíva a szoftver szabadalmakat is lehetővé tevő és ezzel a szabad szoftvereket ellehetetlenítő változatának elfogadását akadályozó munka ügyében szükséges külföldi utazások, és a kapcsolódó hazai rendezvények finanszírozása,
- Az egyesület főállású alkalmazotti bérének egy részének finanszírozására,
- A fennmaradó részt működési tartalékként tettük félre.

Emlékeztetőül, az eddigi években a következő 1% felajánlásokat kaptuk:

- 2002. évben, első alkalommal 225.067.- Ft
- 2003. évben 1.549.026.- Ft
- 2004. évben 2.061.032.- Ft

A támogatás Egyesületünkhöz történő irányításához szükséges nyomtatvány PDF formátumban elérhető az alábbi címen:
http://www.lme.hu/szabalyzatok/1sz_2004.pdf

Egyesületünk pontos megnevezése:
Linux-felhasználók Magyarországi Egyesülete,
adószámunk: 18095647-2-41

Köszönjük, ha 2004. évi személyi jövedelemadójának 1%-át ez évben is nekünk ajánlja fel.

Tisztelettel:
Balsai Péter elnök

Mi újság a rendszermag fejlesztése körül?

A változatkezelési háború még mindig nem csitult el teljesen, de azért már csillapodik. Ebben nagy szerepe van annak, hogy a *BitKeeper* lényegesen jobban működik mint bármelyik ingyenes vetélytársa, valamint, hogy a rendszermag fejlesztők képesség kérelmei különös figyelmet élveznek a fejlesztésben. 2004 novemberében mindössze kétszer került szóba valamilyen alternatív megoldás. Először *Andrea Arcangeli* újra bemutatta a *tla*-t, más néven *arch*-ot, és bár úgy tűnt több más fejlesztő is figyelemmel kíséri az eszköz fejlődését, az általános nézet szerint még nem skálázható megfelelően ahhoz, hogy a rendszermagba kerülhessen.

Később, egy teljesen másik vita folyamán, felmerült a *darcs* változatkezelő kérdése. Karbantartója, *David Roundy*, bemutatta a rendszermag tárház *darcs* tükrét. Ez ugyanis a változatkezelő rendszerek Szent Grálja, hiszen a rendszermag történetének csak a tükre is igen méretes és nehezen kezelhető. Egyelőre azonban a *BitKeeper* stabilan vezet.

Mindig jó új dokumentációt látni. *Alexander Viro* mostanában készült el a C*ross-compilation HOWTO*-jával. Mint mondja, egyáltalán nem nehéz vagy időrabló dolog a rendszermagot az egyik architektúrán fordítani és egy másikon használni. Ő maga általában a következő hat architektúrán szokta ellenőrizni a fordítási hibákat: *i386*, *x86_64*, *sparc32*, *sparc64*, *Alpha* és *ppc*. *Geert Uytterhoeven* készített pár foltot, amelyekkel mindezt az m68k architektúrára is kiterjeszthetjük, de elképzelhető, hogy ez egy ideig még pihenni fog, hiszen a jelenleg az m68k architektúrát nem túl jól támogatja a hivatalos rendszermag forrás. A fejlesztők néhány igen kemény problémával kerültek szembe, amelyekre egyelőre nem találtak megoldást.

A 2.6 rendszermag fejlesztési modell továbbra is *Linus Torvalds/Andrew Morton* karbantartói páros által végzett átalakításokon és tisztázáson megy keresztül. Az új *Signed-off-by* tag például, új lehetőségeket rejt. *Linus* mostanában szeretné egyértelműsíteni, hogy ha foltok mennek ide-oda a fejlesztők között, jobb ha a fejlesztők inkább saját *Signed-off-by* tagjüket helyezik rá, ahelyett, hogy minden egyes küldeményük után megtartanának egy-egy másolatot. Azt is javasolta, hogy minden változás bejegyzéshez csatoljanak egy *CC* listát, bár, mint mondta, ez nem hivatalos dolog, kizárólag információs célokat szolgál és nincs követendő műszaki útmutató. Nos, majd meglátjuk meddig tart mindez. Ugyanakkor elég nagy csapat fejlesztő elégedetlenkedik amiatt, hogy mint *Adrian Bunk* mondotta, *„A 2.6 jelenleg inkább fejlesztői mint stabil rendszermag."* Ezek az emberek úgy gondolják, a 2.7 fejlesztői rendszermagot a lehető leghamarabb el kell ágaztatni, lehetővé téve a 2.6 stabilizálását. Bár a fejlesztés egyértelműen jobb móka mint a karbantartás, *Alan Cox* mégis többször is felajánlotta hogy átveszi a 2.6 karbantartását, márpedig ő híresen nagy hangsúlyt fektet foltjai stabilitására. A 2.6-ac foltsorozata jelenleg rendszermagfejlesztők széles tábora számára jelenti a mértékadó rendszermagot. Egyelőre se *Linus*, se *Andrew* nem mutatta jelét, hogy szándékukban állna *Alannek* átadni a 2.6-ot, vagy lassítani szeretnének fejlesztésük ütemén. Sőt, a 2.6 fejlesztése mostanában épp ellenkezőleg, inkább gyorsulni látszik.

Mindeközben, Linus a kritikák hatására tovább finomítja verziószámozási szokásait. Jelenleg úgy gondolja, hogy a 2.6 sorozat esetében minden előzetes kiadást *rc* jellel lát el, így a legutóbbi az 2.6.10-rc3 jelet kapja. Mindezek mögött ugyanazt az egyszerűségre való törekvést sejthetjük, ami a 2.5-ös az összes *-pre* és *-rc* jelzés elvetésére indította, így végül minden esetben önálló 2.5.x verziót adott ki. A stabil/fejlesztői rendszermag váltakozásának lazulásával, nem tudhatjuk vajon a 2.5-ös szabvány megmarad-e a 2.7-ben is. Egyelőre, az egyetlen biztos pont a rendszermag fejlesztői modell tekintetében az, hogy nincs biztos pont. A *Linux* fejlesztői modell éppen átformálódik, kísérleti stádiumba lépett, felülvizsgálódik, átalakul és bárhogy is végződik a dolog, biztosan egészen más lesz mind ez idáig volt.

A 2.4 rendszermag tovább küzd a stabilitásért. *Marcelo Tosatti* többször felülvizsgálta kitűzött céljait, a teljes lezárástól kezdve néhány fontos új képesség elfogadásán kívül minden más elutasításán keresztül, egészen egy általánosan engedékenyebb elfogadási rendszabályig. Amióta a 2.6 kijött, megpróbálta „mélyfagyasztani" a 2.4-et, de így, hogy a 2.7 nem látható a láthatáron, elég nehéz képességeket elutasítani a 2.4-ből. Mindig vannak akiknek a 2.4 stabilitására van szükségük, de frissebb képességekkel; így aztán egyre több és több dolog kerül át a 2.6-ból, amelyek aztán megpróbálnak utat találni maguknak a 2.4-es forrásfájába.

A *Device Mapper* nemrég utasították el végleg, ugyanis a 2.4-be illesztéséhez túl nagy változtatásokra lett volna szükség, ellenben az *iswraid* (kisebb zökkenőkkel), éppen átcsúszott a rostán. A 2.4.28 után Marcelo újabb elkeseredett kísérletet tett a palack bedugaszolására, bár azért hozzátette *„Az új meghajtók még jöhetnek, feltéve, hogy nem törik meg a meglévő beállításokat és jelentős mennyiségű felhasználó nyer a felvételükkel. Az új meghajtókat pedig olyasvalakinek kell majd elbírálnia, aki az adott területen komolyabb tudással rendelkezik"* – mondta.

Zack Brown

Linux Journal 2005. március, 131. szám

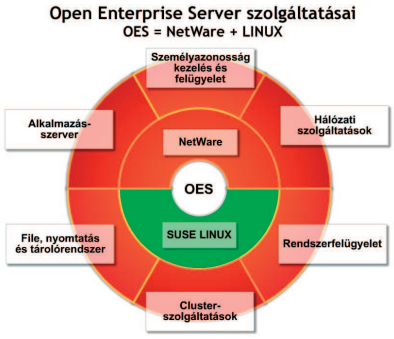
A világ első vegyes forráskódú vállalati rendszere: a Novell Open Enterprise Server

A Novell NetWare legújabb változata, az Open Enterprise Server egyesíti a nyílt és zárt forráskódú rendszerek legjobb tulajdonságait.

Március 3-án a *Novell Magyarország* bemutatta az *Novell Open Enterprise Servert (OES)*, amely a világ első „vegyes forráskódú" vállalati rendszere. Az *OES* egy olyan biztonságos és megbízható hálózati szolgáltatáscsomag, amely a *NetWare* és *SUSE LINUX Enterprise Server* kombinálásával fejlett fájlmegosztási, nyomtatási, felügyeleti, csoportmunka és alkalmazáskiszolgáló szolgáltatásokat nyújt.
Az *Open Enterprise Server* egyesíti a vezető kereskedelmi és nyílt forráskódú hálózati platformok előnyeit és emellett egységes felügyeleti eszközöket, személyazonosság alapú szolgáltatásokat és a *Novell* teljes támogatási rendszerét kínálja a vállalati szintű számítástechnikai igények kielégítéséhez.
Az *Open Enterprise Server* minden idők legtöbbet tudó *NetWare* és egyben *Linux* kiszolgálója. A termék az összes főbb hardver- és lapkagyártó cég – például az *AMD*, *Dell*, *HP*, *IBM* és *Intel* – támogatását élvezi már a piacra kerülés időpontjában.
Az új rendszer fő szolgáltatásai:

- Alkalmazáskiszolgáló
- Fájlkiszolgáló, nyomtatókiszolgáló és tárolórendszer
- Klaszter szolgáltatások
- Rendszerfelügyelet
- Hálózati szolgáltatások
- Személyazonosság kezelése és biztonság

A *Novell OES* a *Linux*, a *NetWare* és a *Windows* felhasználóinak is kínál előnyöket.
Komoly értéket jelent azon cégek számára, akik *Linux* kiszolgálóikat a *Linux* rendszerekben manapság szokványos szolgáltatásokon túl szeretnék bővíteni, hiszen ezeket újabb lehetőségekkel egészíti ki.
Elérhetünk különböző munkacsoportos végfelhasználói hálózati szolgáltatásokat, megvalósíthatunk személyazonosság alapú biztonságot, nagyobb skálázhatóságot érhetünk el, hatékony felügyeleti eszközök állnak rendelkezésünkre, és kapunk egy a beállítást megkönnyítő központi címtárat és minta alapú telepítési lehetőséget is.
Nagy környezetekben mindez jelentősen csökkentheti a beállítással, telepítéssel és üzemeltetéssel kapcsolatos költségeket.



A *Novell* a *NetWare* rendszermagot is továbbfejlesztette az új *NetWare* változatban, így az *Open Enterprise Server* többek között hatékonyabb szerverelérést, fürtözést, protokollkezelést és hardvertámogatást biztosít.
A termék címtárszolgáltatásai biztosítják a kapcsolatot *Active Directory* és tartomány-alapú rendszerekhez egyaránt, az azonosítási, fájl és nyomtató szolgáltatások pedig a megszokott módon teszik lehetővé a felhasználók számára a hálózat elérését.
Bár a Microsoft már nem támogatja a sokak által még használt *Windows NT* rendszereket, az *Open Enterprise Server*-re való átállás számukra is könnyen megvalósítható, költséghatékony megoldás lehet.
A *Novell* által biztosított eszközök megkönnyítik a *NetWare*, *Windows* vagy *Linux* felhasználók számára a *Novell Open Enterprise Server* fejlett szolgáltatásaira történő áttérést.
A felhasználóbarát áttérési és konszolidációs eszközök egyszerű és bonyolult szerverkörnyezetekben is leegyszerűsítik és felgyorsítják az adatok régi kiszolgálókról új kiszolgálókra történő átvitelét. A legfontosabb, hogy az új szolgáltatások párhuzamosan használhatók a már meglévő platformokkal.
A *Novell Open Enterprise Server* tavaly december második felében került a bétatesztelési fázisba, és mintegy 7000 résztvevővel a *Novell* történetének legnagyobb béta-letöltési hullámát indította el. A vezető vírusvédelmi és biztonsági másolatkészítő szoftverek gyártói – a *BakBone Software*, *CommVault Systems*, *McAfee*, *Syncsort*, *Trend Micro* és *VERITAS* – hamarosan olyan termékeket mutatnak be, melyek kiterjesztik az *Open Enterprise Server* képességeit.

Pingvinek a bíróságon

Mostanában egyre többször kapnak szárnyra olyan hírek, hogy önkormányzatok, kisebb-nagyobb szervezetek, cégek szeretnének átállni a ***Linux*** operációs rendszerre. Sajnálatos módon viszont szinte semmit sem hallani arról, hogy az átállás hogyan, milyen körülmények között zajlott, illetve arról sem, hogy azután mik a tapasztalataik az új rendszert illetően, mennyire vannak megelégedve a felhasználók az új rendszerrel, vagy egyáltalán csak, hogy mik a tapasztalataik a Linux-szal kapcsolatban. Sokan nem csak az átállás kihívásaitól félnek hanem attól is, hogy utána milyen kompatibilitási problémákkal kell szembenézniük a mindennapi munka során, megtalálnak-e minden felhasználói programot ***Linux*** alatt is. 2001 októberében már egyszer megjelent egy cikk, hogy a ***Vas Megyei Bíróságon Linux*** operációs rendszert használnak többféle feladatra, úgy gondoltam, hogy érdemes lenne utána járni annak, hogy mi a jelenlegi helyzet, felidézve kicsit az átállás történetét is. Pontosan ezért kerestem meg ***Dologh Ervint***, aki a ***Vas Megyei Bíróság*** informatika osztályvezetője.

– ***Gondolom szerver oldalon már régóta használtatok Linux operációs rendszereket, amikor először felvetődött a gondolat, hogy szabad forráskódú szoftvereket használjatok a munkaállomásokon is. De tulajdonképpen mennyi idő telt el a gondolattól a megvalósulásig, hány számítógépen van Linux operációs rendszer, illetve ez a teljes géppark hány százaléka?***

– Nem egészen így volt. Szervereken már valóban '96-tól használunk ***Linuxot***, klienseken azért nem, mert egyrészt az akkori géppark jelentős része még nem volt „Linux-képes", másrészt nem volt elérhető szövegszerkesztő grafikus felületre. '98 végén, amikor beindult az informatikai fejlesztés, már lehetett „magyarított" ***Applixot*** szerezni. Ez a magyarítás annyiból állt, hogy a menürendszer nagyrészt magyar feliratokat tartalmazott, és az előállított szövegben helyesen jelentek meg az ***Ő*** és ***Ű*** betűk. Nekünk ez akkoriban elegendő is volt, de egy tenderen ilyen programmal nem lehetett volna befutni. Tehát a gondolat már régen megvolt (az én hivatali gépemen már '94 óta van ***Linux***, néhány évvel később pedig már csak Linux).
A megvalósítás pedig lényegében azonnal megtörtént, mihelyt arra lehetőség (pénz) volt. Nagyon fontos tényező, hogy nálunk nem volt szükség átállásra valamilyen régi rendszerről az újra, mivel a régi rendszer DOS-os gépei minden további nélkül illeszkedtek az újba. Persze azt is lehet mondani, hogy a 'megvalósulás' ma is tart, a rendszert folyamatosan bővítjük, alakítjuk, ahogyan azt a lehetőségeink megengedik és a környezeti feltételek megkívánják. És akkor a számok: 2005 január elsején 153 kliens gépünk tud ***Linuxot*** futtatni. Azért nem mondhatom, hogy ennyi gépen van Linux, mert diszk nélküli számítógépeket használunk, és egy-két helyen még szükség van natív DOS betöltésére. Ezen felül van még 1 db ***Windows 95***-ös gép, amin kizárólag a ***GIRO*** rendszer programja fut (ennek nincs linuxos változata), valamint egy ***Windows 3.1***-es gép, amit ***LAN***-nal még el nem látott tárgyalóban szoktak használni. A százalékokat hadd ne számolgassam...
A szerverek terén már változatosabb a kép, a 11 db linuxos szerver mellett működik még egy ***AIX/RS6000*** és egy ***NetWare*** szerver, valamint néhány hónapja 4 db ***Windows 2000*** és egy ***Windows 2003 Server***. Ez utóbbiakra a központilag bevezetendő szoftverek miatt lesz szükség. Az ***AIX***-en az országos bírósági hálózat levelezőrendszere (***Lotus Domino***) működik, bár mi elsősorban a saját ***smtpd-postfix-qmail***-jeinket használjuk, a ***Domino***-ból csak kivesszük a belső leveleket. A ***NetWare*** pedig az e-directory miatt szükséges, noha mi a saját ldap szerverünkről hitelesítünk.

– ***Ha bárkit megkérdeznék arról, hogy vajon miért döntenek szervezetek a szabad szoftverek mellett, a kereskedelmi szoftverekkel szemben, akkor nyilván mindenki azt válaszolná, hogy a szoftverek árának különbsége miatt. A Vas megyei bíróságon is ez volt a legfőbb szempont? Milyen egyéb szempontokat vettetek még figyelembe?***

– Igen, a szoftver ára nagyon fontos szempont. Egyrészt azért, mert ténylegesen ugyanazért a pénzért többet (sokszor lényegesen többet) lehet megvalósítani, másrészt azért, mert az informatikához kevéssé értők is kapnak valamilyen összehasonlítási lehetőséget. Persze itt is be kell tartani a sorrendet: előbb igazolni kell a megfelelőséget, és utána rámutatni az árkülönbségre. Nálunk hasonló volt a helyzet, azzal a különbséggel, hogy a bíróság elnökének látatlanban meg kellett bíznia a bíróság informatikusában. (A bíróság informatikusának pedig abban, aki megígérte, hogy jó lesz az ***Applixware*** :))
Szintén a költség oldalon jelentkezik, de már áttételesebben, hogy lehetőség nyílik a hardver-erőforrások jobb kihasználására. Eleve vékony klienses úton indultunk el, mondván, hogy gyorsan amortizálódó eszközre lehetőleg minél kevesebbet költsünk, a pénzünkért tartós értéket kapjunk. Ezért a hálózatra és a szerverre összpontosítottunk, a kliens gépekből pedig kivettünk mindent, amit lehet. Így nem pazaroltunk erőforrást például az operációs rendszer N példányban való tárolására (a lokális merevle-

mezen lényegében csak az lett volna, esetleg swap partíció), viszont gigabites kábelezést készíttettünk, és tisztán switchelt hálózatot. Ez 1998-ban még nem volt olyan természetes, mint manapság.
Később aztán lettek egyéb hasznos eredői is a választásnak. Egy évvel ezelőtt be tudtunk szerezni elegendően nagy teljesítményű szervereket ahhoz, hogy a szoftverek futtatását a kliensekről átterheljük a szerverekre. Ezáltal jelentősen csökkent a hálózati sávszélesség-igény, de ami még fontosabb, hogy a régi, egyébként már selejtezésre érett gépek terminálként újra használhatók lettek, majdnem olyan használati értékkel, mint az újabbak. Illusztrációként: egy 7 éves ***AMD K6/300***-as gépnél a klikkeléstől számítva 4 másodperc alatt elindul az ***OpenOffice***, a gyorsindító funkció nélkül! Természetesen ha már többen is használják a szövegszerkesztőt, ez az idő még tovább csökken a memóriába olvasás idejével. A ***Megyei Bíróság*** szombathelyi épületeiben a felhasználók asztalán „virtuálisan" 64 bites duál-opteron gép ketyeg. Bírósági viszonylatban azért ez nem az átlagos szint.
A harmadik fontos dolog az, hogy önállóan tudunk új képességeket adni a rendszerhez. Például nálunk már régen 'alanyi jogon' járt az e-mailezés lehetősége mindenkinek, aki accounttal rendelkezett, amikor 'hivatalosan' még csak megyénként 14 db licencet osztottak ki a bíróságoknak egy olyan levelezőrendszerhez, amelyet azóta már ki is dobtak. De van aktuálisabb példám is. A már említett, bevezetni kívánt ***Windows***-alapú szoftvereket nálunk a felhasználók dedikált szerveren fogják terminálos üzemben futtatni. Ezeket a szervereket ezért tűzfallal le tudjuk választani a belső hálózatról, csak a terminál- és az adatbázis elérést engedélyezve. Ráadásul 1 db terminálszerver licenc feleannyiba kerül, mint egy ***OEM Windows XP***, tehát a megnövekedett biztonság mellé bónuszként az olcsóbb bővíthetőséget kapjuk.
És végül, de nem utolsó sorban meg kell említenem a linuxos közösséget, amely főleg a kezdeti időkben igen komoly támaszt jelentett.

– ***Az általad felsorolt szempontok mennyire valósultak meg? Tehát az átállás beváltotta-e a hozzáfűzött igényeket? Esetleg van olyan amit nem?***
Szerintem az elvárt dolgok bejöttek, sőt, az idő múlásával egyre inkább megmutatkozik a nyílt forráskód stratégiai előnye. Persze senki ne gondolja, hogy küszködés, kínlódás, harc nélkül lehet külön úton járni. Sajnos folyamatosan történnek környezetünkben olyan dolgok, amelyek hallatán az informatikában járatos, ám gyanútlan személy rögtön a kollégák otromba tréfájára gyanakszik. Pedig tréfáról szó sincs. Néhány negatív hatású intézkedést éppen a Linux segítségével sikerült közömbösíteni.

– ***Az emberek általában félnek a változástól, és nehezen vehetőek rá új dolgok használatára, aki nem mozog annyira otthonosan az informatikában, annak gondolom még nagyobb nehézséget okozott az átállás. Hogyan támogattátok a felhasználókat átállás előtt, közben, és utána?***
Ebben megint csak szerencsénk volt. Nem volt szükség komolyabb áttérésre, mert a korábbi rendszer elsősorban ***DOS-Clipper*** alkalmazásokat és ***WinWord 2.0*** szövegszerkesztőt adott a felhasználóknak. A ***DOS***-os programokat továbbra is ugyanúgy lehetett használni, a szövegszerkesztőt pedig amúgy is frissíteni kellett volna. Nem akarok senkit sem megbántani, de nálunk nem voltak ***WinWord*** bűvészek vagy Excel zsonglőrök, akik egér nélkül használták volna a gépüket. Az új rendszer a régihez képest gyorsabb volt, mindenki színes monitort kapott, és megmutogattuk, hogy a szükséges műveleteket hogyan lehet elvégezni, az új menürendszerben mi hol van.
A tanulási folyamat az éles használat közben is folyamatosan tart, a felmerülő speciális formázási igényeket együtt próbáljuk megcsinálni, így jobban is rögzül.
A kezdeti applixos idők után már komolyabb feladat volt a ***StarOffice*** bevezetése, de szerencsére a két rendszer békésen megfért egymás mellett, így mindenki a saját maga számára kedvező időpontban válthatott.
Azoknál, akik írógép mellől ültek a számítógéphez, maga a számítógép, annak kezelése jelentett problémát. Az újonnan felvett dolgozók többsége ma már úgy érkezik, hogy használt már számítógépet, az esetek többségében persze valamilyen ***Windows*** verziót. Eleinte némelyik reklamál, hogy nálunk is miért nem az van, meg, hogy miért nincs floppy meg CD olvasó a gépben, de ők is hamar megnyugtathatók.

– ***Sajnos általános probléma, hogy a Linux operációs rendszerekhez nincs annyi felhasználói szoftver, ezért esetleg vannak-e olyan feladatok, melyeket nehézségek árán lehetett megoldani, illetve meg sem tudtátok oldani őket?***
– A bíróságokon használt szoftvereket három csoportba sorolhatjuk aszerint, hogy mennyire általános célúak. Vannak olyanok, amelyek bármely irodában megtalálhatók, például szövegszerkesztő-táblázatkezelő-stb. csomagok, vannak szűkebb (például jogászi, pénzügyi) körben használatos programok, és vannak kimondottan bírósági célszoftverek. Az első csoporttal nincs különösebb gond, kompatibilitási problémák ugyan akadnak, de azokkal együtt lehet élni, esetenként lehet ilyen vagy olyan közbülső adatformát találni. Elvileg a harmadik csoporttal sincs gond, hiszen csak azt kellene mondani, hogy olyan szoftvert kell csinálni, ami multiplatformos, vagy eleve platformfüggetlen. Jelenleg azonban az eredeti elképzelésekkel szemben tisztán ***Win32***-es szoftverek készülnek, inkább adtak mellé szervereket is, hogy legyen min futtatni... Úgy tervezzük, hogy a fejlesztési hullámverések csillapultával itt helyben megkezdjük a szoftverek linuxos megfelelőjének elkészítését. Visszautalva az egyik korábbi kérdésre: ez is egy járulékos előny, rengeteg eszköz van a kezünkben a saját szoftverek elkészítéséhez. A második csoporttal van igazán baj. Ezeket a szoftvereket jelenleg szinte csak ***Win32*** platformra készítik, a ***Kerszöv CD-Jogtára*** egy szabályt erősítő kivétel. Sok esetben nem is tudni, hogy miért nem publikus az ilyen szoftverek forráskódja, hiszen sokszor a mellé csomagolt, rendszeresen frissített adatok a lényegesek. Az, hogy egy magáncég ilyesmire nem figyel, még elfogadható. De egy közcélokat szolgáló állami intézménynél, mint amilyen például az ***APEH*** vagy a ***KSH***, kötelesek lennének figyelni a versenysemlegességre. Hogyha majd elektronikusan kell adatokat szolgáltas-

sunk, mi lesz? Vagy ezek az intézmények jogosultak arra, hogy előírják, hogy milyen rendszert kell használni az ügyfeleiknek?

– ***Ahogyan említetted, hogy legkisebb megyeként nehéz külön úton járni, bennem rögtön felmerült az, a kérdés, hogy mennyire tudtatok így beilleszkedni a bíróságok rendszerébe? A mindennapi munka során mennyi plusz költséget jelent, hogy dokumentumokat, kapott fájlokat konvertáltok Linux alá, majd vissza, illetve, hogy wine-t használtok?***

– Először is: ***wine***-t nem használunk. Egyszer-kétszer próbáltuk, de valahogy nem az igazi. Ami pedig a kétféle rendszer össze-nem-illeszkedését illeti: régebben több volt ebből a gond, manapság már kevesebb. Álláspontom akkor is az volt, ma is az: ha a kirakós játék két darabja nem passzol össze, azért nem lehet csak az egyik vagy csak a másik darabot hibáztatni. Az összekapcsolódás egy kölcsönösséget feltételező folyamat. A konverziókkal kapcsolatosan a magunk részéről mindig valamilyen szabványos formában igyekszünk továbbítani anyagokat, és a partnereinktől is ezt várjuk. Ha olyan ***Excel*** táblát kapunk pl., ami agyon van makrózva, ráadásul jelszóval le is védik, akkor a küldőt megkérjük, hogy ezt vegye ki. Ebből általában még nem volt probléma. Abból már igen, hogy az általunk ***CSV*** formában küldött táblázatot sehogyan sem tudták beolvasni ***Excelbe***, mert a mezőelválasztó nem az alapértelmezett ',' karakter volt, hanem a '|'... A lényeg és a tapasztalat mindenesetre az, hogy kölcsönös jó szándék mellett a problémák megoldhatók. A ***Linux*** nem egy földöntúli csoda, lehet olyan feltételeket előírni és az utolsó vesszőig megkövetelni, amelyeket ***Linuxszal*** ma még nem lehet teljesíteni. Ez módot ad arra, hogy hatalmi szóval a ***Linux*** létalapját meg lehessen semmisíteni. De ha valahol ez a helyzet, ott más, komolyabb baj is van, ott ez már csak tünet. Bármilyen informatikai rendszert is használnak valahol, biztos, hogy a használat közben felmerülnek kisebb-nagyobb gondok, problémák. Ha egy cégnél úgy döntenek, hogy áttérnek nyílt forráskódú rendszerre, nem kevesebb lesz a gondjuk, hanem más jellegű. Megszűnik a rettegés a vírusos levelektől, cserébe alkalmanként molyolni kell a dokumentumkonverzióval. Lényegesnek tartom azonban, hogy a felmerülő problémák megoldása kisebb költséget de komolyabb hozzáértést kíván, és nagyobb döntési szabadságot ad.

– ***Gondolom sok tapasztalatotok gyűlt össze az átállást követően, van esetleg valami amit mindenképpen másképp csinálnál?***

– Nem, azt hiszem nem. Az induláskor a szövegszerkesztő kivételével a többi szoftverkomponenst már letesztelte, a saját munkaállomásom és a már linuxos szerver segítségével ki tudtam próbálni a működőképességet. Innentől kezdve már csak a matematika és a szukcesszív approximáció szabályait követve számolgatni kellett, hogy a rendelkezésre álló pénzből hány munkahelyes hálózat jön ki. A felhasználók részéről nem számítottam ellenállásra, nem is volt. A bíróság elnöke pedig rábólintott. (Később emiatt többször kerülhetett kényelmetlen helyzetbe, nem könnyű legkisebb megyeként külön úton járni.) Az igazsághoz hozzá tartozik, hogy Veszprém megyében is hasonló rendszer készült el és működött évekig, de végül más kézbe került az informatika irányítása és megfelelő gondoskodás hiányában a linuxos rendszer szép lassan eltűnt. Bővüléskor pedig már szinte magától ment minden. (Na jó, azért voltak izgalmas pillanatok :)) Ha N darab gépre kaptunk pénzt, akkor rögtön konvertáltuk 2N darab CD-től, merevlemeztől, floppytól operációs rendszertől megfosztott gépre, esetleg lézernyomtatóra, mostanság lapos monitorra, mikor mire volt éppen szükség. A szállító céggel jó kapcsolatot sikerült kialakítani. Mi is jól járunk, a cég is többet kap (nem sokkal, mert nincs miből) mert a konverziónál mindig marad egy kis hiány.

– ***Hogyan érdemes nekikezdeni egy ehhez hasonló átállásnak, mit javasolsz azoknak a szervezeteknek, vállalatoknak, akik hasonló lépést fontolgatnak?***

– Nyilván nem lehet egy általános sémát adni az áttérésre. Mindenesetre az biztosnak látszik, hogy az informatikusoknak és a vállalat vagy szervezet vezetőinek együttes akarata szükséges. Az informatikusoknak nyilván érteniük kell a ***Linuxhoz***, a vezetőknek pedig – és ez az, ami hihetetlenül nehezen megy – be kell látni, hogy ami ingyen van az lehet jó, sőt, jobb, mint amit pénzért kaphat. Személyesen tapasztaltam több esetben, hogy a döntéshozóval való megbeszéléskor már nem volt egyéb érve a nyílt forráskóddal szemben, csak annyi: Ezt nem hiszem el. Ez nem lehet igaz. Ebben valami trükk van. Ez így nem működhet. Az áttérés: küzdelem. Küzdelem a rögzült szokások, nézetek, hiedelmek megváltozásáért. Ha van közvetlen anyagi nyereség (például amúgy is frissíteni kellene operációs rendszert, szövegszerkesztőt, de ezt szinte ingyen oldjuk meg szabad szoftverekkel), akkor ez erőt adhat, ezért számomra ésszerűnek tűnik a frissítés és az áttérés összekapcsolása. Hasonlóan, ha valahol új rendszer épül ki, rögtön nyílt forráskódú rendszerrel indulni könnyebb, mint menet közben váltani. Jelenleg a közszférában persze automatizmusok vannak, az illetékes döntéshozók fejében föl sem merül ennek lehetősége. Végül is egyszerűbb adóztatni, mint felvállalni egy szokatlan újdonságot.
Ha azonban a fogadókészség a vezetők részéről is megvan, ha nem is sima, de nagyobb buktatók nélküli lehet az út. Erre jó példa a ***Novell***, ahol példamutató módon tették meg ezt a lépést. Könnyű nekik – mondhatnánk – hiszen az egész cég tele van informatikusokkal, nekik ez semmi.
– Tévedés. Egy csökönyös informatikus rosszabb tíz buta adminisztrátornál. Másik példám a Fővárosi Bíróság amelyek fő tevékenysége nem számítástechnikai jellegű, tehát az ottani informatikusok nem a dolgozók derékhadát alkotják, és a vezetők körében az egy főre jutó informatikai ismeretek mennyisége érthető módon alacsonyabb mint a ***Novell*** esetében. De az informatika felkínálta a nyílt forráskód nyújtotta előnyöket, a vezetés pedig nem X vagy Y szoftverkereskedő cég 'szakértőiben' bízott meg, hanem saját embereiben, és – a lehetőségeket mérlegelve – belátta, hogy hosszú távon ez a járható út. Nincs zsákutca, nincs kidobott pénz, nem kell a milliófféle licenc kalkulálásával, nyilvántartásával, frissítésével kínlódni.
Így hát nekiláttak, és én sok sikert kívánok a munkájukhoz!

Az interjút készítette Horváth Ernő

Új termékek

BladeRunner Cluster-in-a-Box

A *Penguin Computing* bejelentette *BladeRunner Cluster-in-a-Box* kiszolgáló rendszerét, mely penge kiszolgálókat, *Ethernet* kapcsolókat, tároló alrendszereket, felügyeleti szoftvert és fürtkezelő operációs rendszert fog össze egyetlen 4 egység magas házba. A *BladeRunner* fürtre előtelepítik a *Scyld Beowulfot*, ez egy kifejezetten fürtkezelésre fejlesztett terjesztés, segítségével egyetlen pontból végezhető el a telepítés, a bejelentkezés és a felügyelet.

A mester csomópontban kettő darab 2,4 GHz-es *Xeon LV* processzor található, 2 GB PC2100-as *DDR RAM*, valamint egy 60 GB-os, 2,5"-os IDE merevlemez. A 11 darab szolga gépben szintén két-két *Xeon LV* processzor és PC2100-as *DDR RAM* üzemel, ám ezek *PXE*-képes, merevlemez nélküli csomópontok. A BladeRunner rendszerek további 4 egység magas házak hozzáadásával és a beépített *Ethernet* kapcsolók csatlakoztatásával akár 42 egységnyi, 240 processzort tartalmazó összeállítássá bővíthetők.

www.penguincomputing.com

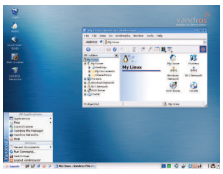
Outblaze-SME

Az *Outblaze-SME* egy értéknövelő viszonteladók számára tervezett elektronikuslevél-kezelő alaprendszer, célközönségét a kis- és középvállalkozások (*small- to medium-sized enterprise= SME*) alkotják. Az *Outblaze-SME* felügyeleti és együttműködést segítő eszközei révén az *SME*-k tárolóhelyet vásárolhatnak, illetve webes felületen keresztül elektronikus levelezési, naptárkezelési és fájltárolási szolgáltatásokat érhetnek el. Együttműködést segítő eszközei lehetővé teszik, hogy a munkatársak naptárakat, névjegyeket és fájlokat osszanak meg egymással, az SME rendszergazdák pedig maguk vehetik kézbe a felhasználói fiókok, a tárolóhely, a csoportlisták és a közös címtárak felügyeletét. Az *Outblaze-SME* a *POP3*, az *IMAP4* és az *SMTP* protokollt egyaránt támogatja, illetve a levelek elérését webes felületen keresztül is biztosítja. Az *Outblaze-SME*-hez az *Outblaze Sentry* nevű vírusvédelmi és levélszemétszűrő szolgáltatása is igénybe vehető.

www.outblaze.com

Xandros Desktop 3

Asztali és hordozható gépekre egyaránt elérhetővé vált a *Xandros Desktop OS 3*-as változata. A harmadik kiadás a 2.6.9-es *Linux* rendszermagra épül, és a *KDE 3.3* testreszabott változatát foglalja magába. A legújabb kiadás újdonságai



között a *Xandros File Manager* alatt fogd és vidd jelleggel végezhető *DVD*-írás, a *Xandros Personal Firewall*, az *Intel Centrino* termékcsalád és vezeték nélküli hálózati csatolóinak támogatása, a felhasználói fájlok önműködő titkosítása, a *PPTP VPN*-ek biztonságos elérése, a *CrossOver Office 4.1*-es változata és a *Xandros Networks* frissítéseire a figyelmet önműködően felhívó értesítők említhetők.

A *Xandros Desktop 3*-as változata lehetővé teszi, hogy a felhasználók fogd és vidd módszerrel bárhonnan bárhová másoljanak, mozgassanak fájlokat, ide értve a windowsos hálózati megosztásokat az *FTP*-helyeket is. Ugyancsak a felhasználók kényelmét és biztonságát szolgálja az önműködő vírusvédelem és levélszemétszűrés is.

www.xandros.com

Kiwi T1x0

Az *EmperorLinux Kiwi T1x0* jelzéssel új, a *Sony T140*-es, *T150*-es, *T160*-as és *T170*-es *VAIO* modelljeire épülő munkaállomásokat mutatott be. A mindössze másfél kilogramm súlyú hordozható gép 1280×768 képpont felbontású, 10,6"-os, szélesvásznú, az X által natívan kezelt *LCD*-képernyővel rendelkezik. A *Kiwi T150 Fedora, Red Hat Enterprise, Debian, Slackware* és *SuSE* minősítéssel rendelkezik. Mindegyik *Kiwiben* 1,1 GHz-es, 2 MB gyorsítótárral ellátott *Pentium-M* 733 processzor található, továbbá 512-1024 MB RAM, 40 GB-os merevlemez és *CDRW-DVD*- vagy *DVD-RW*-meghajtó. Az összes *Kiwi* teljes értékűen támogatja az 1280×768 képpontos, 24 bites színfelbontású módban futó X-et. A lapkakészlet *i855gm*; a hálózati kapcsolatok létrehozására 10/100 Mbps sebességű vezetékes *Ethernet* csatoló, 11-54 Mbps sebességű 802.11 a/b/g *Wi-Fi Ethernet* csatoló, *USB* 2.0 és *IEEE 1394 FireWire* vezérlő használható; a bővíthetőséget *CardBus* foglalat, a kényelmet pedig az *ACPI* alapú hibernálási lehetőség garantálja. A *Kiwi* minden változatához jár az *EmperorLinux* ajándékcsomag, valamint az ingyenes, telefonon és elektronikus levélben igénybe vehető támogatás.

www.emperorlinux.com

DiskOnChip H1

Az *M-Systems* új *DiskOnChip* családot mutatott be, a zenei és mozgóképkezelő készülékekbe szánt termékvonal tagjai akár 8 GB-os tárolókapacitással is rendelkezhetnek. Az elsőként megjelenő 4 GB-os *DiskOnChip H1* 90 nm-es *MLC NAND Flash* és *x2* technológiával készül, az *M-Systems TrueFFS Flash* fájlrendszerét használja, mellyel egyetlen lapkán is lehetségessé válik nagy mennyiségű *MP3* vagy egyéb multimédiás fájl kezelése.

A *DiskOnChip H* sorozat hagyományos, *NOR*-kompatibilis csatolófelülettel rendelkezik, így bármilyen mobil lapkakészlettel használható. A *H1* minden fontosabb – *Symbian OS, Windows Mobile, Palm OS, Nucleus* és *Linux* – mobil operációs rendszert támogat, és minden jelentősebb általános vagy multimédiás célú processzorral képes együttműködni.

m-systems.com

Tervezetkezelés a WebCollab segítségével

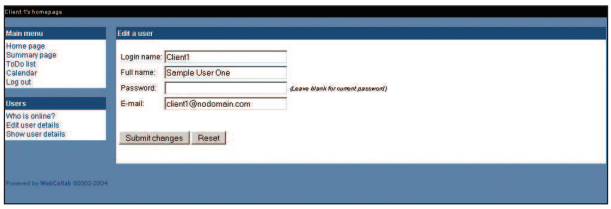
Nyílt forrású, még felfedezésre váró gyöngyszem, mely a tervezetek állapotadatait és a hozzájuk kapcsolódó fájlokat könnyedén áttekinthető, webes felületen teszi elérhetővé. Próbáljuk ki, vajon illeszkedik-e munkavégzési szokásainkhoz!

Hálózati tanácsadóként sokszor dolgozom együtt kisebb csapatokkal informatikai tervezeteken. Bármilyen jellegű is a munka, mindig számtalan kérdést kell egyeztetnem más tanácsadókkal, az ügyfelekkel, adatkereskedő cégekkel és az irodai munkatársakkal. Folyamatosan szükségem van valamilyen megoldásra, amely révén naprakész információkkal tudom ellátni munkatársaimat, és ez sokszor fájlok, feljegyzések megosztásával jár. Korábbi munkahelyemen megtanultam, hogy a ***Microsoft Exchange*** nyilvános mappái és a ***Microsoft Project*** együttese kiválóan alkalmazhatók erre a célra.

Bár a ***Microsoft*** termékei szinte minden számomra szükséges lehetőséget biztosítanak, vannak bizonyos korlátaik. Először is, az ***Exchange/Project*** párosítás egy zárt megoldás, tudását elsősorban a ***Microsoft Windows*** használói tudják kiaknázni, és ők is inkább akkor, ha azonos szervezetbe tartoznak. Másodszor, a ***Microsoft Exchange*** csomagja a legtöbb kisvállalkozás számára túlságosan drága. Harmadszor, bár az ***MS Project*** és az egyéb Gantt-grafikonkészítő alkalmazások de facto szabványnak tekinthetők a tervezetkezelés területén, inkább nagyobb léptékű munkák átfogó kezelésére alkalmasak, például építkezések levezérlésére. Végül, egyesített fájltárolót és tervezetállapot-kezelést akartam, így a tervezetek végigvitele akkor sem okoz gondot, ha a munkatársak különböző szervezetekből érkeznek.

Teljesen véletlenül botlottam bele a ***WebCollabbe*** a ***SourceForge*** oldalán, és rendkívül megörültem neki. Készítői szerint a ***WebCollab*** egy „az együttes munkát segítő, web alapú rendszer tervezetekhez és tervezetkezeléshez; a ***WebCollab*** könnyen használható, közös munkavégzésre bátorítja alkalmazóit. A program kényelmesen kezelhető, funkcionális megjelenésű és biztonságos, miközben használata a legkevésbé sem megterhelő, és komolyabb grafikus teljesítményt sem igényel.”

Igazolhatom, ez mind igaz. A program készítői a tervezés során a használhatóságot a forma elé helyezték, a felület kiválóan átlátható, használata és működése egyszerű és gyors. A ***WebCollab*** ideális megoldás kisebb munkacsoportok számára, amennyiben bennük az adatok áramlásának jellege viszonylag állandó. Nem egyszerűen egy többfelhasználós tennivalók listája, hanem minden tervezethez hozzárendel egy feladatlistát, határidőket, színkódolt előrehaladás-mérőket, fontossági szint beállításokat, fórumot és fájlfeltöltési részt. Ha valamelyik feladat állapota megváltozik, akkor lehetőség nyílik az érintett felhasználók vagy csoportok elektronikus levélben való értesítésére. A fórumokon folyó viccelődések, a fájlcserék és az állapotjelentő elektronikus levelek összességével a ***WebCollab*** valóban interaktív környezetet teremt a tervezetek kezeléséhez. A vezető használhatja a feladatok kiosztására, majd az alkalmazottakkal folyamatos párbeszédet fenntartva bármely pillanatban tisztában lehet a munka előrehaladtával; eközben egy pillanatra sem kell kilépnie a program keretei közül.



1. ábra A rendszergazdák erről az oldalról hozhatnak létre új felhasználói fiókokat. A felhasználó az itt megadott címre kapja az állapotváltozásokról értesítő elektronikus leveleket.

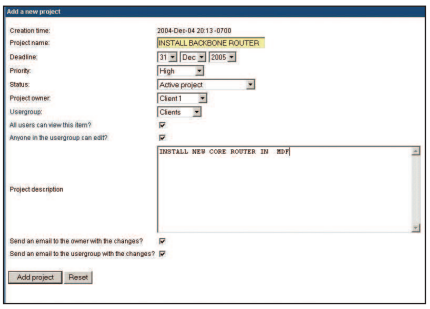
A ***WebCollab*** kínál néhány nagyszerű szolgáltatást, emellett telepítése is egyszerű, használatát pedig gyakorlatilag nem kell tanulni. Kisebb irodai környezetben vagy különböző szervezetekbe tartozó munkatársak részvételekor nagyszerű választás. Én például a ***WebCollab*** segítségével tájékoztatom ügyfeleimet a nekik végzett munkám folyamatáról, de rajta keresztül tartom a kapcsolatot a velem együttműködő mérnökökkel és technikusokkal is.

Rendszerkövetelmények és felépítés

A program felhasználói felülete ***Apache*** és ***PHP*** alapú, háttérrendszere pedig egy adatbázis. Ha a ***PHP*** alapú oldalak elérhetőkké váltak a webkiszolgálón keresztül, akkor böngészőprogram és megfelelő jogosultságok birtokában bárki hozzáférhet a ***WebCollabhez***.

Szerintem a legjobb összeállítás a ***Linux – Apache – MySQL – PHP***, de bármely az ***Apache*** és a ***PHP*** futtatására alkalmas operációs rendszer megfelel, és a ***MySQL*** helyett ***Postgrest*** is választhatunk. Ha szükséges, az adatbázist egy másik kiszolgálóra is helyezhetjük. Jelenleg egy 500 MHz-es órajelű

Pentium III processzorral, 256 MB memóriával és 20 GB-os merevlemezzel felszerelt munkaállomást használok erre a célra, de a körülbelül 15 felhasználó által okozott terheléshez ez alighanem még sok is. A program gond nélkül futtatható egy régebbi, de még munkaképes gépen, illetve meglévő kiszolgálóra is bátran telepíthetjük, nem fog komoly terhelést okozni.
A személyes érzelmeket félretéve is *Windows* vagy éppen *Mac OS* helyett inkább a *Linuxot* javaslom, mégpedig könnyű használata és távfelügyeletének biztonsága miatt. A *WebCollab* telepítéséhez jogot kell szereznünk adatbázis létrehozására, illetve a *WebCollab* könyvtárán belüli jogosultságok némelyikének megváltoztatására. Attól függően, hogy mekkora fájlfeltöltési forgalomra számítunk, néhány száz MB-nyi pluszterületet biztosítsunk webes könyvtárunkon belül. Az egyes fájlok mérete legfeljebb 2 MB lehet.
Az *Apache* és a *MySQL* üzembiztosságával nincs semmi gond, és általában véve a *PHP* kód minőségében sem találtam kivetnivalót. Az viszont tény, hogy vállalati szintű használatra a *WebCollabet* nem ajánlanám. Felhasználói tábora viszonylag kicsi, komolyabb terhelés alatt még nem bizonyított. A legtöbb vállalat a támogatásra is nagy hangsúlyt fektet, amikor kiválasztja alkalmazásait. A *WebCollab* jelenleg egy apró, nyílt forrású tervezet, így nem számíthatunk arra, hogy a nap bármely pillanatában rendelkezésünkre állna egy tetszőleges helyzetet megoldani képes programozó.



2. ábra Az új tervezet létrehozását segítő űrlapon a hozzáférés-vezérlési és a felhasználók értesítésével kapcsolatos beállításokat egyaránt megadhatjuk – egyszerűen csak be kell jelölnünk a megfelelő négyzeteket

Telepítés

A telepítés végletesen egyszerű, nekem tíz percig sem tartott. A terjesztésemhez tartozó alap *Apache*-ot, *PHP*-t és *MySQL*-t tettem fel, ezek tökéletesen működtek is. A *Linux* kezelésében járatlanok számára a legnehezebb a *MySQL* alatti új felhasználó létrehozása lesz, akinek megfelelő hozzáférést is kell adni az adatbázishoz. Ettől eltekintve a csomag telepítése még egy kezdő *Linux*-felhasználó számára sem jelenthet gondot.
A telepítést kétféle módon végezhetjük el, parancssoros vagy webes felületen keresztül. Jómagam ez utóbbit választottam. Példáimban a collab.pelda.com címet fogom használni.
Töltsük le és bontsuk ki a .tar csomagot a webes könyvtárunkba:

```
# tar -zxvf WebCollab-1.62.tar.gz
```

Módosítsuk a fő beállító fájlra vonatkozó engedélyeket:

```
# cd WebCollab-1.62/config
# chmod 666 config.php
```

Böngészőnkben nyissuk meg a *collab.pelda.com/WebCollab-1.62/setup.php* oldalt. Az oldal végigvezet a telepítés önműködő részén, ami magában foglalja egy *SQL* adatbázis létrehozását, a táblákat létrehozó parancsfájl lefuttatását, továbbá négy környezeti változó beállítását a megfelelő *config.php* fájlban.
Állítsuk vissza a fő beállító fájlra vonatkozó engedélyeket:

```
# chmod 664 config.php
```

Felhasználók és csoportok

A program bármely részének eléréséhez felhasználónevet és jelszót kell megadni, és ettől függ, hogy mely tervezeteket tekinthetünk meg és melyeket módosíthatunk. Amint más rendszerekben is, csak a felügyeleti felhasználóknak van joguk a felhasználói fiókok hozzáadására vagy törlésére. Tervezet vagy feladat tulajdonosaként tetszőleges felhasználót kijelölhetünk, ekkor ő kap jogot az adott tervezettel vagy feladattal kapcsolatos felügyeleti teendők elvégzésére.
A csoportok szintén fontos szerepet játszanak a feladatok kiosztásában. A tulajdonos adott tervezetet egy felhasználóhoz vagy egy egész csoporthoz rendelhet hozzá. A tervezeten belüli tennivalókat később alcsoportok vagy feladatcsoportok alkalmazásával lehet kiosztani.
A hozzáférés-vezérlés mindenre kiterjedő, kellő rugalmasságot biztosít a rendszergazdáknak és a tervezetek gazdáinak. Én például sokszor dolgozom együtt más mérnökökkel, és általában mindenkinek jogot kell adnom a feladatok szerkesztésére vagy készre jelölésére, valamint a határidők módosítására. Ilyenkor mindenkinek, aki a saját csoportomba tartozik, szerkesztési jogot adok a tervezetre. Vannak olyan munkáim is, amikor az ügyfeleim csak betekintési engedélyt kapnak, nekik csak olvasási jogot szoktam adni. Természetesen azt sem szeretném, ha az ügyfeleim más munkáimba is belelátnának. A minden felhasználóra vonatkozó megtekintési (***All users can view this item?*** jelölőnégyzet) és a felhasználócsoportba tartozókra érvényes szerkesztési jog (***Anyone in the usergroup can edit?*** jelölőnégyzet) megvonásával mindkét problémát könnyedén meg tudom oldani.

Tervezetek és feladatok létrehozása

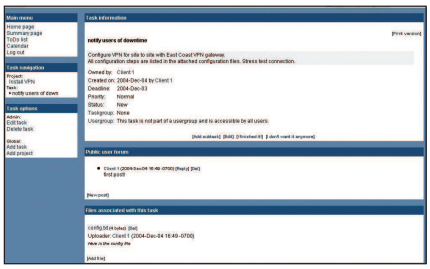
A *WebCollab* főként kisebb, egy vagy két mondattal leírható részfeladatokra bontható tervezetek kezelésére alkalmas. Minden tervezetnek van egy leírása, kezdő és záró dátuma, fontossága és végrehajtó csoportja. Amikor egy tervezeten belül létrehozunk egy feladatot, a program azt egy altervezetként kezeli, egyébként módosítható adatmezői a szülőmunka mezőinek tartalmát veszik át. A programnak ez a része rendkívül hasonló a népszerű tennivalók listája alkalmazásokhoz, ám van annyira rugalmas, hogy nem csupán gyorsellenőrző listaként szolgál, de a feladatok mélyebb részletezését, megjegyzésekkel való ellátását is lehetővé teszi.
A tervezet és a feladat nézetek azok, ahol a program valóban megmutatja a tudását. Korábban már említettem, hogy saját fájlfeltöltő részlege és fóruma van. Itt válik nyilvánvalóvá, hogy a *WebCollab* csoportmunkát segítő szolgáltatásai messze túlmutatnak a hagyományos tennivalók listája alkalmazások tudásán. A felhasználók kérdéseket tehetnek fel egymásnak, megjegyzéseket tehetnek, kapcsolódó dokumentumokat tölthetnek fel stb. Ez az, ami a *WebCollabet* megkülönbözteti a feladatkezelő programocskáktól, és tervezet alapú csoportmunka alkalmazássá nemesíti. A fórum révén a dolgok még interaktívabbakká válnak, és talán a munkatársak figyelmét is jobban megragadják.

Nézetek és navigáció

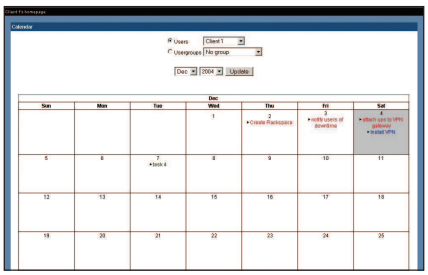
A nézetek közötti navigálásra könnyű ráérezni. Szinte minden elem egy hiperhivatkozás, amelyre rákattintva az adott információt részletesebben is meg tudjuk jeleníteni. Adott tervezet fő nézetében például minden feladatot a címe képvisel, a címre kattintva az adott feladat fő nézetébe jutunk, ahol látható a leírása, határideje, a vele kapcsolatos fájlok listája stb.

A tervezet vagy feladat leírásán belül megjelenő felhasználó- vagy csoportnevek szintén hivatkozások, ezekre kattintva részletesebb adatokat kaphatunk róluk. Alapesetben minden rövidített nézetben jelenik meg, bővebb információkat kattintással érhetünk el. Bár első látásra ez egy kicsit furcsának tűnik, hamar meg lehet szokni, és a nézetek közötti váltás rutinmozdulattá válik. A navigációt a képernyő bal szélén folyamatosan látható sáv is segíti.

Amint a tervezetkezelő csomagoknál jellemző, számos különböző nézet közül választhatunk, ezek mindegyike valamiben eltérő nézőpontot képvisel. A felhasználók bejelentkezés után először a honlap (*Home Page*) nézetet látják. Itt azok a tervezetek és feladatok jelennek meg, amelyekben az illető érintett, valamint látható befejezettségi állapotuk és határidejük is. Két további fontos nézet a tennivalók és a naptár nézet; ezek használata, úgy hiszem, magától értetődő. Ismétlem, a nézetek értelmezése a legkevésbé sem bonyolult, ellentétben a hasonló programok egy részénél megszokottal. Az összes nézetet felhasználó és csoport szerint egyaránt lehet szűrni, és mindegyikhez tartozik nyomtatási nézet gomb is, melyre kattintva az aktuális tartalmat papírbarátabb formában jeleníthetjük meg.



3. ábra A feladat nézetek a kapcsolódó dokumentumokra vezető hivatkozásokat is tartalmaznak. A VPN feladat információs oldalának segítségével könnyen megtalálhatjuk a szükséges beállító fájlt.



4. ábra A naptár nézet alapszintűnek mondható, ám kiválóan szemlélteti, hogy mit mikorra kell elvégezni

Biztonság

A *WebCollab* elsősorban kisméretű vagy otthoni irodai környezetben állja meg helyét. Nem kereskedelmi, nyílt forrású fejlesztés, ezért támogatást hozzá gyakorlatilag csak a weboldalon üzemelő fórumon kaphatunk. Ez hiába tud hatékony megoldás lenni, az öltönyösök az ilyesmit általában nem szeretik. Szerény felhasználói bázisát tekintve kétlem, hogy komolyabb támadások célpontja lenne, ám az *Apache*-ra és a *MySQL*-re már ez nem mondható el. Aki nincs bensőséges viszonyban az *Apache*-csal, a *PHP*-vel és a *MySQL*-lel, az legalább terjesztésének frissítőeszközével ismerkedjen meg, mint a *Debian apt-getje* vagy a *SuSE YaST*-ja, és próbálja minél naprakészebben és minél nagyobb biztonságban tartani ezeket a csomagokat.

A program saját felhasználókezelő és hozzáférés-vezérlési megoldásainak segítségével pontosan szabályozhatjuk, hogy ki mit lát és mit nem, és a szabályozás kellően finom ahhoz, hogy felhasználó, csoport vagy alcsoport szinten is használhassuk. Minden fiókot és jelszót *MySQL* alatt tárol, a jelszavakat természetesen titkosítva. A webkiszolgálót érdemes úgy beállítani, hogy az oldalt *SSL* felett, a 443-as kapun keresztül tegye elérhetővé, amivel akadályozhatjuk a hozzáférések után szaglászó idegenek tényekedését.

Aki az üzletvitel szempontjából nélkülözhetetlen adatokat akar a *WebCollabre* bízni, annak javaslom, hogy dolgozzon egy kicsit a *PHP* és a *MySQL* közötti kapcsolattartás módszerén, különös tekintettel az adatbázishoz tartozó felhasználónév és jelszó átadásának módjára. Összességében ki merem jelenteni, elégedett vagyok a fejlesztők munkájával, amennyiben – a célközönséget is figyelembe véve – a lehető legmagasabb szintű biztonságra törekedtek.

Záró gondolatok

A *WebCollab* szépsége éppen egyszerűségében rejlik. Telepítése és fenntartása könnyű. Átfogó és rugalmas eszköz a kisebb léptékű tervezetek kezeléséhez. Bizonyára vannak, akiket megriaszt az egyszerű felület, ám aki csillogó-villogó oldalak helyett pont ilyen gyorsan megjelenő lapokra vágyik, annak mindenképpen érdemes egy próbát tennie vele. Kipróbált megoldásokra épül, gyors és stabil, és lévén web alapú, használatához nincs szükség külön ügyfélprogramra. A felhasználók hozzáadása és a feladatok, tervezetek létrehozása csupán néhány percet igényel. Időszakos használatra is tudom ajánlani, ha például bárhonnan elérhető tennivalók listájára van szükségünk, hiszen egyfelhasználós módban is remek segédeszköz.

A nyílt forrású közösség újonnan érkezett tagjaként a *Freshmeat* és a *SourceForge* oldalát böngészve úgy éreztem, mintha szabadúszó művészek filmjeit nézném, vagy független rockbandák előadásait hallgatnám. Minden alkalommal, amikor újabb kincsre bukkanunk, úgy érezzük, hogy mi vagyunk az elsők, akik felismerték értékeit – én is így voltam a *WebCollabbel*.

Linux Journal 2005. március, 131. szám

A cikkhez tartozó források elérhetősége:
➲ *www.linuxjournal.com/article/7965*

Mike Cohen az Antropy, Inc., egy kisvállalkozásoknak informatikai tanácsadást végző, Dél-Kaliforniában tevékenykedő cég társalapítója. Szabadidejét a családjával tölti, esetleg szörföl egyet a mexikói Todos Santos sziget közelében.

Optimalizálás a GCC segítségével

Tekintsük át a GCC O kapcsolóinak jelentését, vizsgáljuk meg, hogy bizonyos optimalizálások valójában miért nem azok, aminek gondoljuk őket, valamint hogyan választhatunk alkalmazásainkhoz különleges optimalizálási eljárásokat.

Írásunkban ismertetjük a *GCC* fordító eszközlánc által biztosított optimalizálási szinteket, illetve az ezek által kínált optimalizálási lehetőségeket. Meghatározzuk, hogy mely optimalizálásokat kell explicit módon kiválasztani, ide értve a géptípustól függőket is. Vizsgálatunk elsősorban a *GCC* 3.2.2-es, 2003 februárjában megjelent változatára összpontosít, de megállapításai a jelenlegi, 3.3.2-es változatra is érvényesek.

Optimalizálási szintek

Először nézzük, a *GCC* milyen kategóriákba sorolja az optimalizálásokat, továbbá a fejlesztő hogyan szabályozhatja, hogy mikor melyiket – és sokszor ennél is fontosabb: mikor melyiket nem – kívánja használni. A *GCC* rendkívül sok optimalizálást ismer. Legtöbbjük három szint valamelyikébe tartozik, bár bizonyosak több szinten is elérhetők. Vannak optimalizálások, melyek az eredményként kapott gépi kód méretét csökkentik, mások viszont gyorsabb kódot eredményeznek, akár méretnövekedés árán is. A teljesség kedvéért meg kell említeni a nullás szintet is – explicit módon a `-O` vagy a `-O0` kapcsolóval választhatjuk ki –, melyen semmilyen optimalizálás nem történik.

Az 1. szint (-O1)

Az első optimalizálási szint célja optimalizált kód gyors előállítása. A hangsúly a gyorsaságon van. Az 1. szint két további, sok esetben egymásnak ellentmondó céllal is bír, ezek a kész kód méretének csökkentése és teljesítményének növelése. A `-O1` szint optimalizálásai túlnyomó részt ezeket a célokat szolgálják. Az *1. táblázatban* ezek a `-O1` jelzésű oszlopban szerepelnek. Az első optimalizálási szintet a következő módon engedélyezhetjük:

```
gcc -O1 -o proba proba.c
```

Bármely optimalizálást bármely szinten engedélyezhetünk, ha a `-f` kapcsolóval kísérve megadjuk a nevét:

```
gcc -fdefer-pop -o proba proba.c
```

Megtehetjük azt is, hogy engedélyezzük az első optimalizálási szintet, majd meghatározott elemeit letiltjuk. Erre a `-fno-` előtag szolgál:

```
gcc -O1 -fno-defer-pop -o proba proba.c
```

	Szintek			
Optimalizálás	-O1	-O2	-Os	-O3
defer-pop	●	●	●	●
thread-jumps	●	●	●	●
branch-probabilities	●	●	●	●
cprop-registers	●	●	●	●
guess-branch-probability	●	●	●	●
omit-frame-pointer	●	●	●	●
align-loops	○	●	○	●
align-jumps	○	●	○	●
align-labels	○	●	○	●
align-functions	○	●	○	●
optimize-sibling-calls	○	●	●	●
cse-follow-jumps	○	●	●	●
cse-skip-blocks	○	●	●	●
gcse	○	●	●	●
expensive-optimizations	○	●	●	●
strength-reduce	○	●	●	●
rerun-cse-after-loop	○	●	●	●
rerun-loop-opt	○	●	●	●
caller-saves	○	●	●	●
force-mem	○	●	●	●
peephole2	○	●	●	●
regmove	○	●	●	●
strict-aliasing	○	●	●	●
delete-null-pointer-checks	○	●	●	●
reorder-blocks	○	●	●	●
schedule-insns	○	●	●	●
schedule-insns2	○	●	●	●
inline-functions	○	○	○	●
rename-registers	○	○	○	●

1. táblázat A GCC optimalizálásai és azok a szintek, melyeken engedélyezve vannak

2. táblázat *x86 géptípusok*

Célprocesszor típusa	-march= típus
i386 DX/SX/CX/EX/SL	i386
i486 DX/SX/DX2/SL/SX2/DX4	i486
487	i486
Pentium	pentium
Pentium MMX	pentium-mmx
Pentium Pro	pentiumpro
Pentium II	pentium2
Celeron	pentium2
Pentium III	pentium3
Pentium 4	pentium4
Via C3	c3
Winchip 2	winchip2
Winchip C6-2	winchip-c6
AMD K5	i586
AMD K6	k6
AMD K6 II	k6-2
AMD K6 III	k6-3
AMD Athlon	athlon
AMD Athlon 4	athlon
AMD Athlon XP/MP	athlon
AMD Duron	athlon
AMD Tbird	athlon-tbird

A fenti paranccsal engedélyezzük az első szintet, majd letiltjuk a `defer-pop` optimalizálást.

A 2. szint (-O2)

A második szinten minden olyan az adott géptípuson támogatott optimalizálás megtörténik, amelynél nem kell a sebesség vagy a méret javára dönteni – itt a két szempont kiegyensúlyozottsága jellemző. A hurkok kibontására vagy a függvények helyi kifejtésére például nem kerül sor – igaz, hogy ezekkel a módszerekkel általában növelni lehet a kód sebességét, ám alkalmazásukkor maga a kód is hízik. A második szintet a következőképpen engedélyezhetjük:

```
gcc -O2 -o proba proba.c
```

Az 1. táblázat a `-O2` szint optimalizálásait is tartalmazza. A `-O2` szint a `-O1` szint összes elemét magába foglalja, illetve számos továbbit is tartalmaz.

A 2.5 szint (-Os)

Különleges optimalizálási szint (`-Os`, mint size, vagyis méret), esetében minden a kódot nem növelő, második szintbeli eljárás engedélyezésre kerül. Ilyenkor a hangsúly a méret korlátozására kerül, a sebesség ellenében. Tartalmaz minden második szintű optimalizálást, kivéve a határhoz igazítási (alignment) eljárásokat. A határhoz igazítás azt jelenti, hogy minden függvényt, hurkot, ugrást és címkét olyan címre tolunk el, mely a kettő valamely hatványának többszöröse. Az eljárás géptípustól függő. A határokhoz igazítással a kód és az adatterek mérete és a futtatás sebessége egyaránt nő; ez az oka annak, hogy ezek az eljárások ezen a szinten letiltásra kerülnek. A méretoptimalizálást a következőképpen engedélyezhetjük:

```
gcc -Os -o proba proba.c
```

A gcc 3.2.2-es változata alatt a `-Os` szinten a `reorder-blocks` engedélyezve van, a 3.3.2-es változatnál viszont le van tiltva.

A 3. szint (-O3)

A harmadik, és egyben legmagasabb szint újabb optimalizálásokat tartalmaz (lásd az 1. táblázatot), esetében az elsődleges szempont a sebesség növelése, akár méretnövekedés árán is. A `-O2` szint optimalizálásain túl magában foglalja a `rename-registert` is. Az `inline-functions` (függvények helyi kifejtése) optimalizálást szintén végrehajtja, amivel javul a kód teljesítménye, ám nagymértékben nőhet a mérete is, függően attól, hogy pontosan milyen függvényeket érint a művelet. A harmadik szintet az alábbi módon engedélyezhetjük:

```
gcc -O3 -o proba proba.c
```

Bár a `-O3` szinten gyorsabb kódot kapunk, a méretnövekedés kiolthatja a sebességnövekedés kedvező hatásait. Ha például a kód mérete meghaladja a rendelkezésre álló utasítás-gyorsítótár méretét, akkor számos teljesítménycsökkentő tényezővel kell számolnunk. Lehetséges tehát, hogy a `-O2` szint alkalmazásával jobban járunk, hiszen esetében nagyobb a valószínűsége annak, hogy a kód elfér az utasítás-gyorsítótárban.

A géptípus megadása

Az eddig említett optimalizálások komoly javulást eredményezhetnek az alkalmazás teljesítményében és méretében, ám a célgép típusát megadva további előnyökre is számíthatunk. A gép processzorának típusát a `gcc -march` kapcsolójának segítségével adhatjuk meg. (*2. táblázat*)
Az alapértelmezett géptípus az i386. A GCC minden más i386/x86 alapú géptípuson is működik, ám az újabb processzorok esetében előfordulhat, hogy gyengébb teljesítményt kapunk. Ha fontos a kód hordozhatósága, akkor a fordítást az alapértelmezett beállítással végezzük.
Ha inkább a teljesítmény növelését tartjuk szem előtt, akkor válasszuk ki a gépünknek megfelelő típust.
A géptípus kiválasztásának teljesítményre gyakorolt hatását egy egyszerű példával szemléltethetjük. Készítsünk egy egyszerű próbaprogramot, mely tízezer elemen végez buborékrendezést. A tömbbe az elemeket fordított sorrendben helyezzük el, vagyis a legrosszabb, legtöbb műveletet kívánó esetet vizsgáljuk. A fordítás menetét és a futtatási időket az *1. kódrészlet* ismerteti.
A géptípus megadásával – ez esetben egy 633 MHz-es ***Celeron*** processzorról volt szó – a fordító képessé válik arra, hogy az adott processzortípushoz leginkább illeszkedő utasításokat állítson elő, illetve egyéb, kifejezetten a géptípusra jellemző optimalizálásokat is el tud végezni. Amint az 1. kódrészlet is szemlélteti, a géptípus megadásával a futtatási

3. táblázat *A matematikai egységekkel kapcsolatos optimalizálások*

Beállítás	Leírás
387	Szabványos, 387-es lebegőpontos társprocesszor
sse	Streaming SIMD Extensions (Pentium III, Athlon 4/XP/MP)
sse2	Streaming SIMD Extensions II (Pentium 4)

időben 237 ms-os, vagyis 23 százalékos javulást értünk el. Fontos megjegyezni, hogy az 1. kódrészletben látható sebességnövekedéshez némi méretnövekedés árán jutottunk. A `size` parancs (*2. kódrészlet*) segítségével megvizsgálhatjuk a kód egyes részeinek méretét.
A 2. kódrészletből kitűnik az utasításrész (text rész) 28 bájtos növekedése. Ebben az esetben viszonylag kis árat fizettünk a sebességnövekedésért.

A matematikai egységgel kapcsolatos optimalizálások

Léteznek különleges, az ***i386*** és az ***x86*** géptípusra egyedileg jellemző optimalizálások, melyeket a programozónak kifejezetten ki kell választania, egyébként nem jutnak érvényre. Lehetőség van például matematikai egység választására – igaz, sok esetben ez önműködően megtörténik, a megadott processzor típusa alapján. A `-mfpmath=` kapcsolóval a *3. táblázatban* szereplő egységek közül választhatunk. Az alapérték a `-mfpmath=387`. Létezik egy egyelőre kísérleti jellegű beállítás is, melynél a program az `sse` és a `387` egységet egyaránt megpróbálja kihasználni (`-mfpmath=sse,387`).

Határhoz igazítási optimalizálások

A második szintnél jó néhány határhoz igazítási optimalizálásról volt szó. Ott említettem azt is, hogy ezekkel javítani lehet a teljesítményen, ám méretnövekedést eredményeznek. Ehhez a géptípushoz további három határhoz igazítási eljárás is létezik. A `-malign-int` segítségével a típusokat 32 bites határokhoz tudjuk igazítani. Ha 16 bites igazítású gépre fordítjuk a kódot, a `-mno-align-int` eljárást használhatjuk. A `-malign-double` optimalizálás segítségével a `double`, a `long double` és a `long long` típusokat tudjuk kétszavas határokra rendezni (letiltása a `-mno-align-double` paranccsal lehetséges).
A `double` típusok határhoz igazításával ***Pentium*** gépeken érhetünk el jobb teljesítményt, természetesen a méret rovására. A `-mpreferred-stack-boundary` beállítással a verem igazítására is van lehetőség. Esetében a fejlesztőnek a kettő valamely hatványát kell megadnia a határhoz igazításhoz. Ha például a fejlesztő a `-mpreferred-stack-boundary=4` értéket adja meg, akkor a verem 16 bájtos határhoz igazodik – ez az alapbeállítás is. ***Pentium*** és ***Pentium Pro*** processzorokon a verem `double` változóit 8 bájtos határhoz érdemes igazítani, a ***Pentium III*** processzorok viszont 16 bájtos igazítással teljesítenek jobban.

Sebességnövelés

A szabványos függvényeket – mint a `memset`, a `memcpy` vagy az `strlen` – használó alkalmazások esetében a `-minline-all-stringops` beállítással, a karakterlánc-műveletek helyi kifejtésével tudjuk növelni a teljesítményt. Természetesen mellékhatásként itt is számolnunk kell a kód méretének növekedésével.
A hurkok kibontása úgy történik, hogy a fordító egy-egy ciklusban a lehető legtöbb munkát végezteti el a programmal, így kevesebb ismétlésre van szükség. Ez esetben a teljesítmény javulása és a méret növekedése ismét együtt jár. A hurkok kibontását a `-funroll-loops` beállítással engedélyezhetjük. Azokban az esetekben, amikor az ismétlések számát nehéz meghatározni, márpedig ez a `-funroll-loops` használatának előfeltétele, a `-funroll-all-loops` optimalizálással lehet az összes hurkot kibontani.
Hasznos eljárás a `-momit-leaf-frame-pointer`, ám használata megnehezíti a kód hibáinak felderítését. Segítségével a keret mutatót (frame pointer) egy regiszteren kívül tarthatjuk, így kevesebbszer kell megadni és törölni az értékét. Mindemellett a regiszter elérhetővé válik a kód számára is. A `-fomit-frame-pointer` optimalizálás szintén jó szolgálatot tehet.
A `-O3` szinten, illetve a `-finline-functions` beállítás használatakor egy különleges átadott érték felületen keresztül megszabhatjuk, hogy legfeljebb mekkora függvényeket akarunk helyileg kifejteni. Az alábbi parancsban például a helyi kifejtésű függvények méretét 40 utasításban korlátozzuk:
`gcc -o rendez rendez.c –param max-inline-insns=40`

1. kódrészlet A géptípus megadásának egy egyszerű alkalmazás futására gyakorolt hatása

```
[mtj@camus]$ gcc -o rendez rendez.c -O2
[mtj@camus]$ time ./rendez
real    0m1.036s
user    0m1.030s
sys     0m0.000s
[mtj@camus]$ gcc -o rendez rendez.c -O2 -
➥march=pentium2
[mtj@camus]$ time ./rendez
real    0m0.799s
user    0m0.790s
sys     0m0.010s
[mtj@camus]$
```

2. kódrészlet Az 1. kódrészletben szereplő program méretváltozása

```
[mtj@camus]$ gcc -o rendez rendez.c -O2
[mtj@camus]$ size rendez
   text    data     bss     dec     hex filename
    842     252       4    1098     44a rendez
[mtj@camus]$ gcc -o rendez rendez.c -O2
➥-march=pentium2
[mtj@camus]$ size rendez
   text    data     bss     dec     hex filename
    870     252       4    1126     466 rendez
[mtj@camus]$
```

3. kódrészlet: Egyszerű példa a gprof használatára

```
[mtj@camus]$ gcc -o rendez rendez.c -pg -O2
➥-march=pentium2
[mtj@camus]$ ./rendez
[mtj@camus]$ gprof –no-graph -b ./rendez
➥gmon.out
Flat profile:
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.00      0.79     0.79        1   790.00   790.00  bubbleSort
  0.00      0.79     0.00        1     0.00     0.00  init_list
[mtj@camus]$
```

Ezzel a módszerrel kézben tarthatjuk a `-finline-functions` alkalmazása kapcsán jelentkező kódméret-növekedést.

A kód méretének optimalizálása

A verem alapértelmezett határhoz igazítási értéke 4 vagy 16 szó. Helyszűkével küszködő rendszereknél az alapértéket a `-mpreferred-stack-boundary=2` beállítással nyolc bájtra is állíthatjuk. Állandók, például karakterláncok vagy lebegőpontos értékek megadásakor ezek a független értékek általában saját helyet foglalnak el a memóriában. Jobb, ha nem engedjük szabadjára őket, ugyanis az azonos jellegű állandók összefogásával csökkenthetjük a tárolásukhoz szükséges hely méretét. Ezt a különleges optimalizálást a `-fmerge-constants` beállítással vehetjük igénybe.

Optimalizálás grafikus hardverelemekre

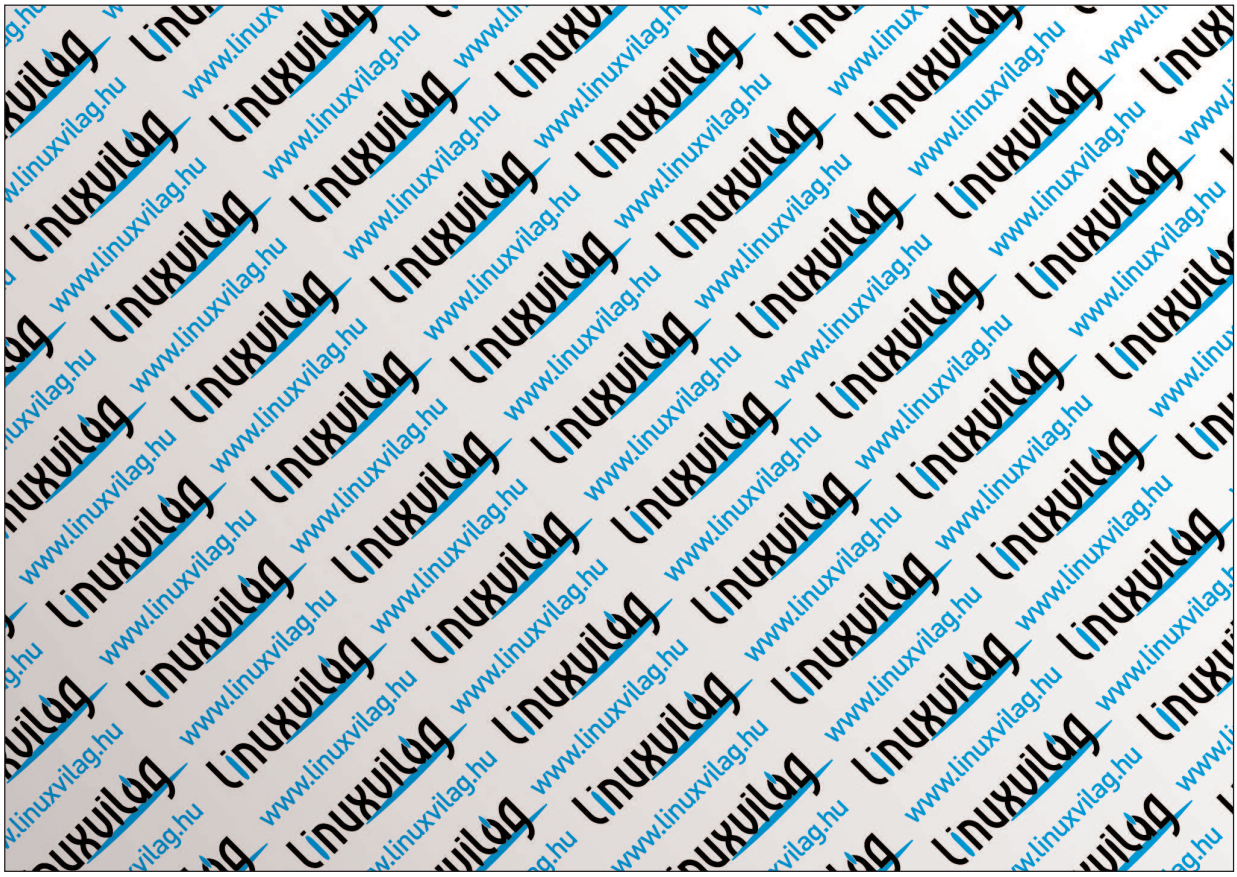
A megadott célgép típusától függően számos további kiterjesztés kerül engedélyezésre. Ezeket explicit módon is lehet engedélyezni vagy tiltani. A `-mmmx` és a `-m3dnow` például önműködően engedélyezésre kerül, amennyiben a megadott processzortípus támogatja az adott utasításkészletet.

További lehetőségek

Számos a sebesség növelésére és a kódméret csökkentésére alkalmas optimalizálásról és kapcsolóról ejtettünk szót, most említsünk meg néhány mellékes, ám sokszor roppant hasznos lehetőséget.

A `-ffast-math` optimalizálás olyan átalakításokat végez, melyek nagy valószínűséggel helyes kódot eredményeznek ugyan, ám nem biztos, hogy szigorúan igazodik az *IEEE* szabvány előírásaihoz. Bátran használhatjuk, ám gondosan teszteljük le az eredményt.

Ha a globális közös alkifejezések kiküszöbölése engedélyezve van (`-fgcse`, `-O2` vagy magasabb szint), akkor



további két lehetőség nyílik a betöltési/elmentési műveletek számának csökkentésére. A `-fgcse-lm` és a `-fgcse-sm` optimalizálás a betöltő és elmentő műveletek hurkon kívülre helyezésével alkalmas a hurkon belül végrehajtott utasítások számának csökkentésére, amivel nő a hurok lefuttatásának sebessége. A `-fgcse-lm` (`load-motion`, betöltő műveletek) és a `-fgcse-sm` (elmentő műveletek) kapcsolókat együtt kell használni.
A `-fforce-addr` optimalizálás arra kényszeríti a fordítóprogramot, hogy a címeket regiszterekbe mozgassa át, mielőtt bármilyen aritmetikai műveletet végrehajtana rajtuk. Hasonló a `-fforce-mem` beállításhoz, mely a `-O2`, a `-Os` és a `-O3` szinten alapesetben engedélyezve van.
A mellékoptimalizálások közül utolsóként a `-fsched-spec-loadot` említeném meg, mely a `-fschedule-insns` optimalizálással együtt használható. A `-O2` szinttől kezdve alapesetben is engedélyezve van. Segítségével mód nyílik bizonyos betöltő utasítások elméletileg hatékonyabb végrehajtást eredményező áthelyezésére, amivel a lehető legkisebbre csökkenthető az adatfüggőségek miatt a futtatásban bekövetkező fennakadások száma.

A kapott javítások kipróbálása

A korábbiakban a `time` paranccsal vizsgáltuk meg az adott utasítások végrehajtására fordított időt. Az alkalmazások profilozásakor természetesen ennél jóval részletesebb betekintést kell nyernünk a kód működésébe. A ***GNU*** `gprof` segédprogram és a ***GCC*** fordító együttesen képesek megfelelni az ilyen jellegű igényeknek is. A `gprof` tárgyalása túlmutatna jelenlegi témánkon, ám a *3. kódrészlet* jól példázza a használatát.
A kódot a `-pg` kapcsolóval fordítjuk le, így belekerülnek a profilozáshoz szükséges utasítások. A kód lefuttatása után az eredmények a `gmon.out` fájlba kerülnek, ebből a `gprof` segédprogrammal állíthatjuk elő az emberi szem számára is olvasható profilozási adatokat. A `gprof` futtatásakor ebben az esetben a `-b` és a `–no-graph` kapcsolókat használtam. Ha tömör kimenetet szeretnénk (vagyis el kívánjuk hagyni a mezők bővebb magyarázatait), a `-b` kapcsolót kell használnunk. A `–no-graph` kapcsoló letiltja a függvényhívási grafikon megjelenítését; ez egyébként az egyes függvények közötti hívásokat és a függvények futtatásának idejét szemlélteti.
A harmadik kódrészletre tekintve megállapíthatjuk, hogy a `bubbleSort`, vagyis a buborékrendezést végző eljárás egyszer került meghívásra, és futtatásának ideje 790 ms volt. Az `init_list` függvényt szintén meghívtuk, ennek futtatása kevesebb mint 10 ms-ot igényelt (ez a profilozási mintavétel felbontása), ezért mellette nullás érték szerepel.
Ha a sebesség helyett inkább a méretváltozások érdekelnek bennünket, akkor a `size` parancsot kell használnunk. Még részletesebb adatokhoz az `objdump` segédprogrammal juthatunk. Például a kódban lévő függvények listáját a `.text` részekre keresve állíthatjuk elő:

```
objdump -x rendez | grep .text
```

A kapott listából ezután ki tudjuk választani a komolyabb érdeklődésünkre is számot tartó függvényt.

Az optimalizálások vizsgálata

A *GCC* optimalizáló lényegében egy fekete doboz. Megadjuk neki a beállításokat és a megfelelő kapcsolókat, majd kapunk egy kódot, ami vagy jobb, vagy rosszabb. Ha javulást látunk, vajon mi történt pontosan? Erre a kérdésre a kód vizsgálatával kaphatunk választ.
A célutasítások kiírására a -S kapcsolóval vehetjük rá a fordítót:

```
gcc -c -S proba.c
```

Ekkor a `gcc` előfordítja a kódot (`-c`, mint compile), illetve megjeleníti a forrás assembly kódját (`-S`). A kapott assembly kimenet a `proba.s` fájlba kerül.
Az előző megközelítés hátránya, hogy csak assembly kódot látunk, a tényleges utasítások méretéről semmit nem tudunk meg. Ha ez a célunk, akkor az `objdump`-pal az assembly mellett natív utasításokat is elő tudunk állítani:

```
gcc -c -g proba.c
objdump -d proba.o
```

Az előfordítást a `-c` kapcsolóval kértük a `gcc`-től, a hibakeresési adatokat pedig a -g kapcsolóval illesztettük be. Az `objdump` a `-d` kapcsoló hatására szedi szét az objektumkódban szereplő utasításokat. Végül az assemblyvel megszórt forrást az alábbi paranccsal kapjuk meg:

```
gcc -c -g -Wa,-ahl,-L proba.c
```

A parancs futásakor a ***GNU*** assembler állítja elő a kódforrást. A `-Wa` kapcsolóval a `-ahl` és a `-L` kapcsolót adjuk tovább az assemblernek, amely így a szabványos kimenetre magas szintű kódból és assemblyből felépülő tartalmat ír ki. A `-L` kapcsoló a szimbólumtábla helyi szimbólumainak megtartását szolgálja.

Összefoglalás

Mivel minden alkalmazás más, nem adható olyan bűvös beállításegyüttes, amellyel minden esetben a legjobb eredményt lehetne elérni. Megfelelő teljesítményt a legegyszerűbben a `-O2` optimalizálási szint használatával kaphatunk. Ha a hordozhatóság nem szempont, akkor a `-march=` kapcsolóval adjuk meg a célprocesszor típusát. Ha tárhely tekintetében szűkösen állunk, akkor elsőként a `-Os` szinttel próbálkozzunk. Ha a lehető legnagyobb teljesítményt akarjuk kipréselni a kódból, akkor több szinttel is próbálkozzunk meg, a kapott kódot pedig vizsgáljuk meg az említett segédprogramokkal. Adott optimalizálások engedélyezésével vagy letiltásával szintén van esélyünk arra, hogy a lehető legjobb teljesítményt hozzuk ki a fordítóból.

Linux Journal 2005. március, 131. szám

A cikkhez tartozó források elérhetősége:
➲ ***www.linuxjournal.com/article/7971***

M. Tim Jones (mtj@mtjones.com)
A longmonti, Colorado állambéli Emulex Corp. vezető főmérnöke. Belsőprogram-tervező mérnök, emellett nemrég készült el BSD Sockets Programming from a Multilanguage Perspective című könyvével. Korábban kommunikációs és tudományos műholdakhoz írt rendszermagokat, jelenleg hálózati készülékekhez fejleszt belső programokat.

Bemutatkozik az Ardour

A Linux rögzítőstúdiónk szíve a merevlemezes rögzítő. Vegyük szemügyre hát az Ardour-t, amely profi szintű rögzítési módszereket hoz a nyílt forrás világába.

A kortárs zenészek számítógépeiket ***digitális audio munkaállomásként (DAW)*** rengeteg, hanggal kapcsolatos műveletre használják. A leggyakoribb műveletek a rögzítés, hangállomány szerkesztés, hatások hozzákeverése, dinamikus feldolgozás illetve az audio számok felkészítése a CD mester lemezre íráshoz.
A modern zenészek számítógép alapú stúdiójában központi szerepet játszik a ***merevlemezes rögzítő (HDR)***. Az ***Apple*** vagy ***Microsoft Windows*** gépeken dolgozók jó néhány ***HDR*** rendszer között válogathatnak, ***Linux*** alatt azonban, egészen a közelmúltig nem volt egyetlen igazán professzionális rendszer sem. Professzionális minőségű ***HDR*** készítése nem éppen egyszerű feladat és az üzleti ***HDR*** fejlesztők nem túl sok technikai segítséget nyújtottak az önjelölt nyílt forráskódú ***DAW*** fejlesztőknek.
Manapság azonban, ***Paul Davis*** vezető tervező/programozónak és tehetséges csapatának hála, a linuxos zenészek rendelkezésére áll a őshonos fejlesztésű és professzionális színvonalú ***Ardour HDR/DAW***.

Mire jó, és mire nem

Az ***Ardour*** magas minőségű hanganyagokhoz szánt, többsávos rögzítő és szerkesztő rendszer. Az ***Ardour*** támogatja az audio feldolgozó bővítményeket (***LADSPA*** és ***VST***) valamint automatizált paraméter vezérléssel, kifinomult ***csúsztatás (panning)*** vezérléssel és sok fejlett szerkesztési funkcióval rendelkezik. Ismeri a ***MIDI időkódot (MTC)***, a ***MIDI SMPTE*** kódolási átlag időkódot, a ***MIDI gépi vezérlést (MMC)*** azaz a ***MIDI*** külső keverők és rögzítők vezérléséhez szánt üzenetkészletét, illetve a ***JACK***-et, azaz a ***Linux*** és ***Mac OS X*** alatt használt alacsony lappangási idejű kiszolgáló és alkalmazás átvitelvezérlés felületet.
Adattípusokat is támogató professzionális hangrögzítési alkatrészeket ritkán találunk a fogyasztói szintű rendszerekben. Egy profi ***DAW*** nagyobb bitmélységben képes kezelni a hangot, a mélyebb amplitúdó tartomány és a nagyobb pontosság érdekében; magasabb mintavételezést alkalmaz, a pontosabb frekvencia felbontás miatt; és nagyobb rugalmassággal rendelkezik a hang- és térkezelés területén. Egyáltalán nem ritka, hogy a profi rögzítők 32-bites, 96kHz mintavételezéssel rögzített hangfájlokkal dolgozzanak, ami több mint kétszerese a kompaktlemezek felbontásának.
Az ***Ardour*** nem ***MIDI*** állomány rögzítő vagy szerkesztő. Semmit nem tud a hangjegyekről, és nem is hangállomány-szerkesztőnek szánták. Az ***Ardour*** csak néhány beépített jelfeldolgozó képességgel rendelkezik, további feldolgozási képességeit bővítményekből meríti. Végül, közvetlenül nem nyújt lehetőséget CD-készítésre és írásra, viszont úgy írták meg, hogy jól együttműködjön a kiváló ***JAMin*** tervezőkészlettel.

1. ábra Az Ardour többsávos rögzítő és szerkesztő rendszer amely egyaránt támogatja a fogyasztói és professzionális alkatrészeket

Alkatrészigény

Az ***Ardour*** használhatóságát nagyban befolyásolja a rendelkezésünkre álló alkatrészek képességei. Amennyiben a hangkártyánkat és hangeszközünket támogatja az ***ALSA*** hangrendszer, valószínűleg működni fog az ***Ardour***-ral is; ugyanakkor a szokványos fogyasztói cikkek nem igazán alkalmasak az ***Ardour*** képességeinek teljes kihasználására. Nagyszerű dolgokat lehet csinálni az ***Ardour*** és egy ***SoundBlaster Live*** párossal is, de profi munka esetében az ***Ardour*** jobban örül egy többcsatornás digitális audio csatolófelületnek, legyen az ***RME Hammerfall*** vagy éppen ***M-Audio Delta*** kártya.
Az auido felületünk megvásárlása előtt érdemes előtanulmányokat végezni. Nézzük meg az ***ALSA*** oldalán megjelenő támogatott alkatrészeket és próbáljunk meg keresni valakit aki megjegyzéseket fűzött a választott kártyánkhoz.
Az ***Ardour*** képes válaszolni a ***MIDI*** paraméter-vezérlésre, így a ***MIDI*** megvalósítás lapot is érdemes áttanulmányozni

azokhoz a külső eszközökhöz amelyekkel használni szeretnénk. Nézzük meg, a keverőnk adatai között szerepel-e az *MMC* vagy *MTC* rendszer. Az *Ardour* képes használni az automatikus keverővezérlést, de akár csak az előbb, az alkatrésznek is alkalmasnak kell lennie ilyesmire.
Az *Ardour* felhasználói levelezőlistán gyakran felmerül az elégséges felhasználói rendszer kérdése. Kielégítő eredményekről számoltak be 500 Mhz-es processzorral is, de egy ilyen rendszer professzionális felhasználáshoz már biztosan nem elegendő. A gyors processzor pontos szinkronizálást biztosít, miközben több sávot rögzítünk vagy játszunk le, és néhány hatás használatához pedig feltétlenül szükséges egy kis sebesség. Például egy jó visszhang hatás igencsak *CPU*-igényes művelet. Ezen kívül egy nagy és gyors merevlemezre is szükségünk lesz, melyet a *hdparm* eszközzel optimális működésre bírtunk. Többsávos, áramló adatok szinkronizálásához viszont feltétlenül gyors *CPU* és lemez szükséges. A 32-bites digitális hangállományok elég nagyok lehetnek és a egy szokásos rögzítési folyamat előre nem látható lemezmennyiséget igényelhet. Profi felhasználáshoz érdemes két lemezzel felszerelni a gépünket. Az egyik lemezt a rendszernek és alkalmazásoknak tartjuk fönn, a másikat pedig kizárólag hangadatok rögzítésére használjuk. Az *Ardour* weblapján különféle ajánlott *CPU* és merevlemez paramétereket találunk, sőt még javasolt alaplapokat is. Ha a legjobb eredményt szeretnénk elérni az Ardour-al, kövessük a tervezők tanácsait.
Aaron Trumm kiváló cikke, a *„The Linux-Based Recording Studio”* foglalkozik a komoly rögzítési feladatokhoz használható külső felszerelések beállítási kérdéseivel. Ahelyett, hogy itt elismételném *Aaron* javaslatait, olvasóinkat inkább arra bíztatnám, hogy e cikkben nézzenek utána a mikrofonok, keverők, monitorok, hangfalak és egyéb külső felszerelésekkel kapcsolatos kérdéseknek.

A program igényei

Az *Ardour* jelenlegi nyílt változatát forráskódként tölthetjük le. Néhány *Linux* terjesztéshez (*Red Hat/Fedora, Mandrake, Debian* és *Slackware*) csomagok is rendelkezésre állnak.
A hálózati források között megtaláljuk őket. A *CVS* elérés jelenleg nem nyilvános, de az *Ardour* weblapján mindig megtaláljuk az aznap esti tarlabdát. Az *Ardour* forrásból történő fordítása nem különösebben bonyolult, és az összes szükséges információt megtaláljuk a tarlabdában.
Az *Ardour* weblapján utánanézhetünk a program fordításához és telepítéséhez szükséges legfrissebb támogató eszközöknek. Az *Ardour* működéséhez szükséges csomagok: az *ALSA* rendszermag hangrendszer, a *JACK* audio-kiszolgáló, a *LADSPA* bővítmény *API* és gyűjtemény valamint különféle audio-vonatkozású programelemek. Meglévő *Linux* telepítésünket is beállíthatjuk úgy, hogy optimális teljesítménnyel fusson a program, de ha komolyan szeretnénk az *Ardourral* foglalkozni, érdemes audio-optimalizált rendszert választani. Ilyenek például a *Debian* alapú *AGNULA/Demudi*, vagy a *Planet CCRMA*, *Red Hat/Fedora* csomagok. A *Slackware* felhasználók *Luke Yelavich AudioSlack* csomagjait telepíthetik, a *Mandrake* felhasználók pedig valamennyi szükséges csomagot megtalálják *Thac* lapján.
Az autófeldolgozást segítő *JAMin* eszközkészlet feladata a mesterlemezre írandó sávok előkészítése. A mesterkészítő folyamatban használt főbb eszközök a tömörítők, ekvalizerek és határolók (limiters) amelyek a sávok közötti amplitúdó és a frekvencia egyenlőtlenségeket csökkentik. Ha egy teljes CD lemezt szeretnénk rögzíteni az *Ardourral*, érdemes a *JAMinnel* készíteni a mestert.

2. ábra Átméretezett sávok, területszerkesztő és az LADSPA bővítmény

Főbb képességek

Minden modern *HDR* a három általános digitális hang munkafolyamatnak megfelelően három alapvető feladatot lát el, azaz: rögzítés, szerkesztés és keverés. Valamennyi fázis önmagában is elég összetett feladat, az *Ardour* praktikus felhasználói felülete mégis könnyen kezelhetővé teszi a bonyolult programot.
Az *Ardour* a rendelkezésre álló alkatrész korlátokig dolgozik. Ha 32-csatornás I/O hangfelülettel rendelkezünk és található hozzá *ALSA* meghajtó egyszerre 32 hangcsatornával dolgozhatunk. A csatornák sávokhoz rendelése nagyon rugalmas, és minden sáv monó illetve sztereó bemenet kezelésére is alkalmas. Az *Ardour* szinkronizálási felülete jelenleg legjobban a *JACK*-rendszerű alkalmazásokkal működik, így például a *Hydrogen* dobgéppel vagy a *Rosegarden audio/MIDI* szekvenszerrel. Rengeteg munkát fektetnek ugyanakkor a *MIDI* időkód fejlesztésbe is.
Mint korában említettük, az *Ardour* nem hangfájl szerkesztő, de azért tartalmaz néhány többsávos rögzítésre kihegyezett szerkesztési műveletet. Ilyen például az általános, és a nemisannyira-általános kivágás/másolás/beillesztés műveletek, csoportosított sávszerkesztés illetve a sáv és szegmensáthelyezések részletes vezérlése. Az *Ardour* maga is képes időnyújtási és amplitúdó normalizálási feladatok elvégzésére, ám a feldolgozás oroszlánrészét az *LADSPA* bővítmény végzi.

Az LADSPA

A *Linux Audio Developers Simple Plugin Architecture (LAPSDA)* lényegében egy könnyen használható programozói felület, melynek segítségével hatásokat és más bővítményeket építhetünk a linuxos audio alkalmazásainkba.
Az *API*-t annyira széles körben használják a *Linux* audio fejlesztők, hogy a felhasználók ma már elvárják, hogy egy új hangkezelő vagy zeneprogram ismerje a *LADSPA* rendszert.
Az *Ardour* adatsávokon, *területeken (regions)*, *tartományokon (ranges)*, *darabokon (chunks)* és *csoportokon* képes dol-

3. ábra A Mixer panel sávcsíkját úgy kell elképzelnünk, mintha a hangadat felülről lefelé mozogna minden egyes sávban

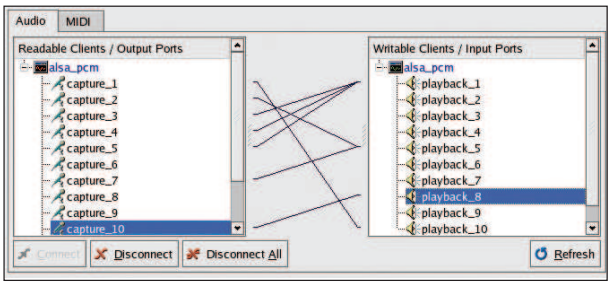
gozni. Az adatmegjelenítés változtatható: nagyíthatjuk vagy kicsinyíthetjük a sáv megjelenítését, a csoportosítás segítségével pedig csak az adott szerkesztési csoportba tartozó sávokat látjuk. A 2. *ábrán* megfigyelhetjük az ***Ardour*** sávmegjelenítőjének átméretezési képességét, a területszerkesztőt és az ***erősítés (gain)*** automatizáló görbét.

Hány sávot használjunk?

Minthogy általában felvételkészítők használják, a sáv kifejezés pontosan megfelel a mágneses szalagra rögzíthető hang fogalmának. A digitális hangot más módszerrel rögzítjük a merevlemezen, ám a régi kifejezés továbbra is megmaradt, hiszen több audiofolyamot grafikusan folyamatos sávként jelenítünk meg. A csatorna fogalmát ugyanakkor talán a keverő működésén keresztül lehet legjobban megérteni. A többcsatornás keverők több bemenetet (csatornát) kezelnek, amelyek jeleit szabadon összevegyíthetik és átirányíthatják, mielőtt azok a keverő kimenetére kerülnénk. Ehhez hasonlóan a többcsatornás digitális hangfelület is több vegyíthető és átirányítható csatornát kezel a ***HDR*** sávjainkon. A csatornák számát a csatolt eszköz határozza meg.
Mint korábban említettük, a fogyasztói termékek ritkán képesek sztereó ki/bemenetnél többre, míg a professzionális felszerelések akár 50 vagy több csatornát is kezelhetnek. A lemezhely gyorsan fogy a felvételeink különféle változatai, ideiglenes sávjai és biztonsági másolatai tárolása során. A program korlátozhatja az használható sávok számát, de az egyszerre rögzíthető sávok számát a merevlemez sebessége határozza meg. Ne feledjük, a magas minőségű digitális hang igen keményen igénybe veszi a rendszererőforrásainkat.
Az ***Ardour*** keverőpanelje sávonként egy-egy csúszkából és mester vezérlőcsíkból áll. A sávcsík a „felülről lefelé” adatlogikának megfelelő szakaszokból áll. A csík tetején kezdődik a bemenet, ez után következnek a vezérlők (jelpolaritás, szóló és néma sáv állapot), aztán keresztülhalad a ***előkioltás (pre-fader)*** bővítményen, magán a ***sáv kioltás (fader)*** szakaszon, majd a ***utókioltás (post-fader)*** bővítményeken, végül eléri végleges állapotát ahonnan a kimenetre vagy kimenetekre kerülhet. A kimenet általában, de nem feltétlenül a mester buszt jelenti. Az ***Ardour*** keverőjének további figyelemre méltó képességei: automatikus erőszabályzás, automatikus rögzítési és visszajátszási kioltás mozgatás; keverőkbe és vissza irányuló ***MIDI***-vezérlés a ***MIDI***-gépvezérlésen keresztül; valamint az automatikus bővítményparaméter kezelés.
Mivel az ***Ardour Erik de Castro Lopo sndfile*** könyvtárát használja, a hangadatokat (***sáv***, ***terület*** és ***munkafolyamat (session)***) a ***libsndfile*** által ismert bármelyik formátumban exportálhatjuk és importálhatjuk. Jelenleg körülbelül 20 különféle audio fájlformátumot használhatunk, beleértve a legnépszerűbb fogyasztói és profi formátumokat is.

Az Ardour munkafolyamatai

A következő munkafolyamat leírásával az ***Ardour*** mint többsávos rögzítő rendszer képességeit szeretném bemutatni. Megpróbáltam kerülni a bonyolult műszaki kifejezéseket, de a cikket nem bevezetőnek szántam a digitális rögzítés világába. A témával kapcsolatos alapinformációkat megtaláljuk a ***www.homerecording.com*** lapon, illetve komolyabb szinten a ***www.prorec.com*** oldalon. Sok egyéb hálózati és nyomtatott forrást találhatunk a ***Google*** és ***Amazon*** megfelelően formázott kereséseivel.
A felhasznált alkatrészek: ***M-Audio Delta 66 digital audio*** felület. Ez a rendszer ***PCI*** kártyával és ***breakout*** dobozzal rendelkezik, amely összesen 4×4 analóg I/O és 2×2 digitális ***I/O*** csatornát biztosít. A digitális kapukat egyaránt állíthatjuk ***S/PDIF*** azaz magas színvonalú fogyasztói szintű digitális ***I/O*** jelre, vagy rögzítési iparban használt ***AES/EBU*** szabványra. Minden bemenet sztereó kapu, így lényegében összesen 12 bemeneti csatornánk van, ami lényegesen rugalmasabb kombinációkat tesz lehetővé mint egy szokásos fogyasztói egyszerű sztereó hangkártya.
A munkafolyamatban két külső keverőt használtam.
A külső szintetizátorokhoz ***Yamaha DMP11*** alkalmaztam, a vokál, gitár és harmonika előadásokat pedig egy ***Tascam TM-D1000*** vezérelte mielőtt a ***Delta 66***-ra kerültek volna.
A ***Tascam*** keverő ***S/PDIF*** digitális kimenettel rendelkezik, ezért a ***Delta*** kártya digitális bemenetére tudtam kapcsolni.
Az volt a tervem, hogy az ***Ardour*** segítségével több hangszer és vokál sávon rögzítem az eredeti dalt, majd keveréssel újra előállítom a kis csapat élő zenéjének hangzását.
Persze ebben a munkafolyamatban a kis csoport egyedül én voltam, kiegészítve néhány importált ***WAV*** állománnyal és egy kis ***Ardourbeli*** többsávosítással. Szerettem volna néhány ***LADSPA*** bővítményt is használni, hogy hatásokkal tűzdeljek meg bizonyos sávokat, valamint ki akartam használni az ***Ardour csúsztatás (pan)*** vezérlőjét amellyel beállíthatom a sávok pontos helyzetét a sztereó audio térben, ahogy a zenészeket állíthatnánk a színpadon.
A zongora, basszus, gitár és dob részekhez ***MIDI*** szekvenszert használtam. Minden részt ***MIDI*** állományként mentettem el, majd mindegyiket ***WAV*** audio állománnyá alakítottam a népszerű ***TiMidity MIDI*** eszközzel. A dobsávot sztereó sávvá, a többit pedig ***monaural*** állománnyá alakítottam, továbbá minden fájl 16-bites 44.1kHz WAV formátumban készült.
Ezek után valamennyi készen állt a ***Ardourba*** olvasásra.



4. ábra A Delta 66 I/O csatornái a qjackctl megjelenítésében

Amikor először megnyitjuk az ***Ardourt***, egy üres sávmegjelenítőt láthatunk, illetve egy udvarias figyelmeztetést, miszerint létre kell hoznunk egy ***Ardour*** munkafolyamatot a ***Session/New*** dialógusban. Munkafolyamat sablonokból is válogathatunk, mivel azonban én tudtam a pontos sávigényeimet inkább saját sávkiosztást hoztam létre egy sztereó és négy monó sávhoz. A ***WAV*** állományokat a sávokba importáltam és már meg is volt a kísérő együttesem.
Következő lépésként további három sávot rögzítettem az ***Ardourban***, felvéve a ritmusgitár részt, a vokál sávot és a harmonika szólót. ***Ardour*** alatt a rögzítés nagyon egyszerű: kattintsunk a kiválasztott sáv R gombjára (ezzel élesítjük felvételre), majd állítsuk be a bemeneteket és a szinteket a sáv keverőcsíkjában. Ezt követően kattintsunk a főablak nagy vörös ***Record*** gombjára, bökjünk a ***transport play*** vezérlőre és máris rögzíthetjük amit szeretnénk. Egy vagy több további sávot is figyelemmel kísérhetünk, és valós időben szóló módba kapcsolva vagy elnémítva a sávokat vagy sávcsoportokat különféle együtteseket tesztelhetünk.
Amikor elégedett voltam a rögzített előadással, elkezdtem dolgozni a keverésen. A nyers keverékben több dologra is oda kellett figyelni: a ritmusgitár sávnak ***hangerőkioltást (volume fade-out)*** kell adni a ***kiegyenlítési (EQ)*** átvitel során, a ***MIDI*** hangszerek nem voltak elég élesek a keverékben valamint minden normalizálni és egyensúlyozni kellett. Szerencsére az ***Ardour*** mindezt igen egyszerűen kezelte. A kiegyenlítéshez és hangosításhoz az ***LADSPA*** bővítményeket, illetve szükség esetén az ***Ardour*** saját belső normalizálási rutinját használtam. A vokál és harmonika részeknél alkalmazott visszhang hatás szintén a ***LADSPA*** bővítmény érdeme.
A ritmusgitár sávom ***kioltása (fade-out)*** érdekes feladatnak bizonyult. Először is a sáv automatizálási gombjára kattintva beállítottam az automatizálási görbét, majd az egérrel és a ***Gain mode*** gomb segítségével meg tudtam rajzolni az amplitúdó görbét. Később felfedeztem, hogy az ***Ardour*** vezérlés automatizálását a keverőben is használhatom. Az automatizálási állapotot írásra állítottam, majd lejátszottam a kioltandó munkarészt és csökkentettem a kioltó szintjét. Miután az ***Ardour*** rögzítette a kioltó változását, az automatizálási állapotot visszaállítottam lejátszásra, és a kioltó lejátszás közben magától lefelé mozdult. Ráadásul a kioltókat szimulációs műveletek (így az együttes automatizálási görbekészítés) esetén is összekapcsolhatjuk keverőcsoportokká.

Normalizálás és kiegyenlítés

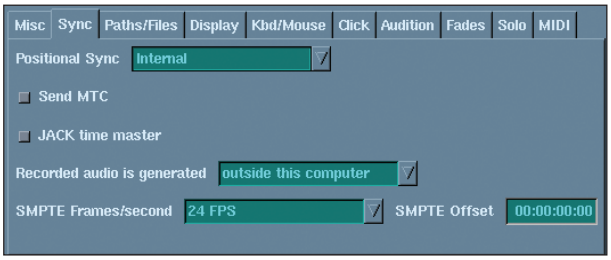
A normalizálás során a jel legnagyobb amplitúdóját a maximumra növeljük vágás előtt. Az összes többi amplitúdó az új csúcsamplitúdó arányában változik. A ***kiegyenlítés (equalization)*** egy adott frekvencia vagy frekvenciatartomány erejét növeli vagy csökkenti. Az autók sztereó hangrendszeréből ismert, úgynevezett ***polcos (shelving) grafikus kiegyenlítő (ekvalizer)*** jó példa az ilyen ***EQ*** eszközre. Ezek az eszközök a hallható skálát többé kevésbé vékony részekre osztják, ahol valamennyi részben lehetőségünk van erősíteni vagy gyengíteni az adott frekvencia tartományt.
Hogyan végződött ez a kis kreatív kísérlet? Mindenki saját fülével győződhet meg róla; az ***URL***-t a források között találjuk.
Ne feledjük, hogy a felvétel még nincs kiírva, és még visszatérhetek az eredeti munkámhoz és módosításokat végezhetek rajta. Szívesen látok minden javaslatot, de azért legyünk elnézőek szerény próbálkozásomhoz.

Szinkronizálás

A ***JACK*** alacsony lappangású audio kiszolgáló és átvitel vezérlő csatolófelület szintén ***Paul Davis*** szerzeménye, így bizton számíthatunk rá, hogy az ***Ardour JACK*** szinkronizálási képességei elég fejlettek. A ***Hydrogen*** dobgépet és a ***Rosegarden*** szekvencert tesztelve a mester és szolga ***JACK*** módokban vegyes sikereket értem el. Mindkét program gond nélkül szinkronizált az ***Ardourral*** oda-vissza ***JACK***-en keresztül. A ***Hydrogen*** kiválóan működött ám a ***Rosegarden*** szoftszintetizátor kimenetének rögzítésével gondjaim voltak. A ***Rosegarden*** csapat ismeri a problémát és tervezik a javítását, így a ***Rosegarden*** weblapján érdemes utánanézni a legfrissebb híreket.
A tesztelés idején az ***Ardour MTC*** küldés/fogadás része éppen komoly újraírás alatt állt, így nem tudtam igazán használható teszteket készíteni. Ugyanakkor az ***MTC*** támogatás egy jelentősebb elem, ami sok ***Ardour*** felhasználó kívánságlistáján megtalálható, így a fennmaradó hibák minden bizonnyal javítva lesznek még az 1.0 megjelenése előtt. Az ***Ardour*** jövendőbeli verzióiban valószínűleg egyéb szinkronizálási lehetőségek is helyet kapnak majd. Várólistán találjuk a ***SMPTE*** olvasást, ***MIDI*** órát és ***SPP***-t ***(song position pointer, azaz számpozíció mutató)*** mint lehetséges jelölteket, de ezekkel a protokollokkal már csak az ***Ardour 1.0***-ás verzió megjelenését követően fogunk tudni dolgozni.

Benyomások

Minél többet tudok meg az ***Ardourról***, úgy tűnik annál többet kell még tanulnom. Az ***Ardour*** tervezőinek hála a munkafelület tiszta és zavaró elemektől mentes, A legördülő és felbukkanó menük további lehetőségeket rejtenek. Ezen kívül rengeteg hasznos gyorsbillentyű segíti munkánkat, felderítésükhöz az `ardour -b` parancsot kell kiadnunk az `xterm` parancssorában.
Miután kicsit magabiztosabban mozogtam az ***Ardour*** alapképességei között, elkezdtem felfedezni a további képességeit. A ***TM-D1000 MIDI*** üzeneteket és vezérlőfolyamokat küld a legtöbb művelet végrehajtásakor, az ***Ardour*** vezérlőjét pedig külső ***MIDI*** vezérlőfelülethez is rendelhetjük.
A gombon vagy kioltón a ***ctrl+középső egérkattintással*** majd a vezérlő aktiválásával tudjuk hozzárendelést elvégezni. Ez a képesség nagyon hasznos, hiszen segítségével bármely, ***MIDI*** vezérlőfolyamokat küldő külső egység fel-



5. ábra Opció ablak

használható ***Ardour*** vezérlőfelületként. Mellesleg, a csoportosítás ilyenkor is érvényes, így az ***Ardourban*** akár több kioltót is kezelhetünk egyetlen külső kioltóval.
A ***TM-D1000*** képes ***MIDI gépi vezérlő parancsokat (MIDI machine control, MMC)*** értelmezni és küldeni, ami igen hasznos képesség, ugyanis az ***Ardour*** képes vezérelni illetve vezérlehető az ilyen eszközökről. Az ***MMC*** üzenetek olyan általános átvitelvezérlési műveleteket irányítanak, mint az indítás/leállítás, gyors-tekerés és visszacsévélés. Így egy ***MIDI***-érzékeny mixer, például a ***TM-D1000***-es, szinte az összes ***Ardour*** művelet vezérlőfelületeként felhasználható.
Egy ilyen terjedelmű cikkben csak a céloknak leginkább megfelelő képességeket tudtam kipróbálni. Még nem dolgoztam az ***Ardour*** ciklus-rendszerével, az ***MTC*** a cikk készítése közben fejeződött be, nem ellenőriztem az időnyújtási képességet sem, és így tovább. Mint említettem, az ***Ardour*** igen nagy tudású alkalmazás, és cikkünkben rengeteg érdekes és hasznos képességet még csak nem is érintettünk.

A jövő

Az ***Ardour*** fejlesztési aktivitása igen viharos, különösen most, hogy a program közeledik az 1.0 -ás kiadáshoz. Sok ember érdeklődik egy versenyképes alternatíva iránt a más operációs rendszereken megszokott üzleti zárt megoldások helyett, és az ***Ardour*** úgy tűnik a helyes fejlesztési ösvényen halad. Még igen sok munkára lesz szükség, ideértve a ***MIDI*** képességek továbbfejlesztését, videó követés támogatását, kiterjesztett szinkronizálást és a ***GUI*** nagyjavítását (***GTK2*** támogatást terveznek). Ugyanakkor jelenleg az ***Ardour*** képesség fagyasztás alatt áll, az elsődleges szempont a hibajavítások és a stabilitás elérése az 1.0-ás verzió kiadása előtt.
Mint azt egy komolyabb összetett projekt ***pre-1.0 béta*** változatától megszokott, ez sem teljesen hibátlan. Az ***Ardour Mantis*** hibakövető és képességkérő rendszere viszont kiváló lehetőséget biztosít egy ismert hiba állapotának ellenőrzésére, új hibák jelzésére, vagy a kívánt képességek iránti igény bejelentésére.
A ***VST*** bővítmények támogatása jelenleg kicsit problémás, ami legfőképpen a ***WINE*** és a ***Linux*** rendszermag folyamatos fejlesztésnek tudható be, de ezt a kérdés is előkelő helyet foglal el a fejlesztők listáján. Sok felhasználó jelezte, hogy hajlandó lenne platformot váltani, ha a ***VST/VSTi*** támogatás zökkenőmentesen működne ***Linux*** alatt.

A VST/VSTi bővítmények

A ***VST/VSTi*** audio bővítmények rendkívül fontosak a hangprogramok világában. A ***VST API***-t a népszerű ***Cubase audio/MIDI szekvencer*** készítője, a ***Steinberg*** cég készítette, amely később világszerte elterjedt a fejlesztők és a felhasználók között. Pontosan fogalmazva a ***VST*** bővítmény általában egy hang vagy ***MIDI*** feldolgozó, a ***VSTi*** bővítmény pedig egy hangszer mint a szintetizátor vagy a dobgép. Manapság több ezer ***VST/VSTi*** bővítmény létezik a házi készítésűektől kezdve a drága üzleti megoldásokig.
Sok, igen jó minőségű ingyenes ***VST*** bővítmény létezik, sőt néhány ***VST*** szerző saját bővítményét teljesen szabad, nyílt forrású programként is kiadta a ***GPL*** engedélye alatt.
Az ***Ardour*** dokumentációja szintén érdekes kérdés, ugyanis jelenleg nincsen hivatalos felhasználói kézikönyve. ***Paul Davis*** a további fejlesztések során is valószínűleg ingyenesen fogja elérhetővé tenni az ***Ardourt***, és csak a jó minőségű kézikönyvért kér majd ellenszolgáltatást. Addig is a merevlemezes rögzítők alap tervezési kérdéseiben nem annyira járatos felhasználóknak érdemes beszerezni és tanulmányozni valamelyik üzleti ***DAW***, például a ***Pro Tools*** vagy a ***Cubase*** dokumentációját. Néhány ***Ardour*** vonatkozású dokumentációt találhatunk a forráscsomag szöveges állományaiban és különböző hálózati forrásokban, például a Quick Toots sorozatban (lásd a forrásokat), valamint az ***ardour-users*** és ***ardour-dev*** levelezőlisták forgalmában. A fejlesztők és tesztelők az ***#ardour IRC*** csatornán beszélgetnek, az egyéb felhasználók ***Ardour***-vonatkozású kérdéseiket az ***AGNULA/Demudi, Planet CCRMA***, ***ALSA*** és ***Linux Audio Users*** csoportban tehetik fel.
Készen áll vajon az ***Ardour*** a nagy dobásra? Talán még nem teljesen, de az útirány egyértelműen arrafelé mutat és a hátralévő út sem lehet már túl hosszú. Úgy gondolom, már csak rövid ideig kell várnunk és az ***Ardour*** az élvonalbeli üzleti audio-program világot is szemöldökfelvonásra készteti majd. Az ***Ardourt*** már ma is használják teljes ***CD*** projektek rögzítésére és keverésére, és egyre több felhasználó számol be arról, hogy sikeresen használta az ***Ardourt*** saját rögzítési projektjében. Úgy tervezem még rengeteg zenét készítek az ***Ardour***-al. Bárki nyugodtan látogassa meg a honlapomat és szemezgessen az alkalmi szerzemények között.

Köszönetnyilvánítás

A szerző köszönetét és mély elismerését küldi ***Paul Davisnek***, ***Taybin Rutkinnak***, ***Jesse Chappellnek***, ***Steve Harrisnak*** és az ***Ardour*** fejlesztőcsapat többi tagjának. Munkájuk az ***Ardourban*** és sok más értékes ***Linux*** audio projektben komoly újítás és igazi munkaszeretetről tanúskodik. A világ szabad zenészei tisztelegnek előttetek!
Nagy köszönet illeti az ***Ardour*** felhasználók levelezőlistáját, különös tekintettel ***Jan Depner***, ***Mark Knecht***, ***Aaron Trumm*** és ***Josh Karnes*** urakat. A fejlesztők és felhasználók nagyon segítőkészek voltak, és amikor elakadtam az ***Ardour*** trükkösebb részeinél a jó társaság igencsak megkönnyítette a nehézségeket.

Linux Journal 2005. március, 131. szám

Dave Phillips az Ohioi Findlay-ben élő zenész, tanár és író. Linuxszal való eslő, 1995-ös találkozása óta tagja a Linux Audio közösségnek. Ő a s zerzője a The Book of Linux Music & Sound című könyvnek valamint Linux Journal számos cikkének.

GNU Motion: A számítógép termek mindent látó szeme

Lehet, hogy jól néz ki az ajtónk, de biztosan szükségünk van 23 órányi videóanyagra ahol zárva látjuk? A következő program használatával biztonsági videóinkból kigyűjthetjük a lényeges részeket és kiszúrhatjuk a be- és kilépéseket.

Tegyük fel, hogy van egy több ezer dolláros számítógép felszerelésekkel teli szobánk. Ez igazán olyasmi amin érdemes rajta tartani a szemünket, nem igaz? Fel is rakunk egy hálózati csatlakozással ellátott kamerát. Ettől kezdve csak a kamera honlapjára kell benéznünk és máris látjuk mi is történik a kiszolgálóteremben éjjel és nappal. Ez már haladás, de hamarosan rájövünk, hogy valamilyen rögzítőeszközre is szükség lenne, hátha esetleg ki kell találni, ki is volt a szobában az utolsó csütörtökön. Ezért aztán elkezdjük menteni a videóanyagokat a hálózat másik rendszerére, hogy aztán szükség esetén visszanézhessük őket. Esetleg írunk néhány parancsfájlt, amelyek mondjuk hetente lecserélik a felvételeket, hogy azok ne töltsék fel a merevlemezt. Miután hosszú órákon át bámuljuk a videót, hogy végül ráakadjunk ki vette „kölcsönön" kedvenc csavarhúzónkat, hirtelen ráébredünk, itt bizony további finomításokra lesz szükség. Hát nem lenne sokkal jobb, ha a számítógép csak az érdekes videórészeket tartaná meg, a többit pedig kidobná? Nos, használjuk a ***GNU Motion***-t, ezt az ingyenes mozgásérzékelő programot. Dolgozzuk fel vele a videóinkat és a napi 24 órás felvételeink 15 perces klippé zsugorodnak, rögzítve minden pillanatot amikor valami megmozdult a szobában – éljen a technika.

Alkatrészigény

A ***GNU Motion*** önálló webkamerákkal is tud dolgozni (ilyeneket árul például a hálózati forrásokban található ***Axis***), illetve használhatunk bármilyen, ***video4linux***-kompatibilis rögzítőkártyához kapcsolt kamerát. Itt most az ***Axis 2100***-as önálló kamerán alapuló megoldásra fogunk koncentrálni, ugyanis ezt egyszerűbb beállítani. Mindkét esetben szükség lesz egy ***Linux*** rendszerre, amely a videó anyagot elmenti és futtatja a ***Motion***-t. A ***Motion*** elég komoly feldolgozóerőt kíván, de egy ***Pentium III*** processzorral rendelkező vagy annál erősebb gép megfelelő lehet, különösen ha kizárólag a ***Motion*** futtatására használjuk.
Az ***Axis*** kamera telepítése és beállítása nem túl bonyolult. Keressünk neki egy jó helyet a megfigyelendő szobában, majd vezessünk be az áramot és az ***Ethernet*** kábeleket. Tapasztalataim szerint valamivel szemmagasság felett, 7 láb magasan a szoba sarkában elhelyezett kamerával érhetjük el a legjobb lefedettséget. Kövessük a kamera telepítési útmutatóját és rendeljük a hálózatunk valamelyik ***IP*** számához. Ellenőrizzük a kamera helyes működését, majd irányítsuk a böngészőnket a kamera weblapjára. A videótárolásra és a ***Motion*** futtatására használt gép tetszés szerinti helyre kerülhet. Az egyszerűség kedvéért talán érdemes a kamerával azonos logikai vagy fizikai hálózatra helyezni.

Programigény

Bármely modern ***Linux*** terjesztés megteszi. Saját megoldásomban ***Fedora Core 1***-et használok.
Töltsük le a ***Motion*** honlapjáról a ***Motiont*** (lásd a forrásokat). Az írás születésekor a 3.1.16-os volt a legfrissebb verzió. Felhasználhatjuk a ***Motion*** weblapján található ***RPM*** csomagot de lefordíthatjuk forrásból is. Máshonnan leszedett ***RPM*** és ***Debian*** csomagok telepítését nem igazán javaslom, ezek ugyanis általában idejétmúltak és hiányzik belőlük egy-két képesség. A ***Motion*** fejlesztésének néhány hónapja alatt elég sok fontos változás történt.
Az egyetlen külső programfüggőség az ***ffmpeg*** könyvtár, amelyet a ***Motion*** az ***MPEG*** videók készítésére használ. Az ***ffmpeg 0.4.8***-as kiadott verzióját kell használnunk, ugyanis az újabb, fejlesztési verziók mellett a ***Motion*** nem működik helyesen. Töltsük le az ***ffmpeg*** forrását (lásd a forrásokat); ***ffmpeg*** könyvtárat a ***Motion*** telepítése előtt kell lefordítanunk és telepítenünk. Ha nem így teszünk a ***Motion*** megpróbálja a régebbi, ***mpegplayer*** nevű eszközzel létrehozni a videókat. Mivel legtöbbünknél valószínűleg ez sincs telepítve, a ***Motion*** nem igazán fog jól működni.

Program fordítás

Miután letöltöttük a ***Motion***-t és az ***ffmpeg***-et, kicsomagoljuk őket például a ***/tmp*** könyvtárba. Lépjünk `cd`-vel az ***ffmpeg*** forráskönyvtárába és adjuk ki a következő parancsokat:

```
$ ./configure
$ make
# make install
```

Az utolsó parancsot root-ként kel kiadnunk.

A parancsok a */usr/local/lib* könyvtárba helyezik el az *ffmpeg* programkönyvtárat. Ezek után lépjünk be a *Motion* forráskönyvtárába és ismét futtassuk le a `./configure` parancsot. Ezúttal azonban ellenőrizzük le az eredményt. Fontos, hogy a *Configure Status*, *FFmpeg Support* mellett *Yes* választ lássunk. Amennyiben nincs így, a *Motion* nem találta meg az *ffmpeg* könyvtárat a rendszerünkön. Ez a *Motion* telepítésekor fellépő hibák és kavarodások elsődleges oka. Ne is folytassuk, amíg meg nem oldottuk ezt a éproblmát. Keressük meg, hogy a rendszerünk hol tárolja a *libavcodec-0.4.8.so* állományt, majd a *Motion* könyvtárban futtassuk le ismét a *configure*-t:

```
$ ./configure --with-ffmpeg=/valamilyen/elérési/út
```

Ha a `configure` futtatása után azt látjuk, hogy `FFmpeg Support: Yes`, akkor végre lefordíthatjuk és telepíthetjük a *Motiont*:

```
$ make
# make install
```

Akárcsak az előbb, az utolsó parancsot rootként kell futtatni. Ha elkészültünk, a */usr/local/bin/motion* végrehajtható állomány a rendelkezésünkre áll.
Amennyiben a telepítés során problémákba ütköznénk, nézzünk utána a *Motion Guide* (lásd a forrásokat) oldalain. A kézikönyv bizonyos részei kicsit elavultak, de hasznos információkat tartalmaz a *Motion* telepítésével és futtatásával kapcsolatban.

Motion beállítása

A *Motion* démonként fut, folyamatosan analizálva és tárolva a videóanyagot. Vezérlését a szokásos *UNIX* mintának megfelelően egy beállításfájl végzi. Másoljuk át a forráskönyvtár *motion-dist.conf* állományát az */etc/motion.conf* helyre, majd szerkesszünk át néhány paramétert. Az első, amit meg kell változtatnunk a `netcam_url` beállítása. A *Motion* ezen az *URL*-en keresztül kapja a *JPEG* képeket a kameráról. Az *Axis 2100* kamera esetében ez a következő alakú lesz:

```
http://netcam.example.com/axis-
cgi/jpg/image.cgi?resolution=640x480
```

Miután a *motion.conf* állományban beállítottuk a `netcam_url` változót, a közvetlenül csatlakoztatott kamerákra vonatkozó beállításokat (videóeszköz, forgatás, magasság, szélesség) figyelmen kívül hagyja a rendszer. Nem árt ha tudjuk, hogy a netkameráknak van egy hátrányuk a szokásos videó felvevő eszközökkel szemben. Jelenleg a *Motion* csak egyetlen *JPEG* képet tud egyszerre lekérni a netkameráktól, ezáltal a videónk maximum 12-15 képkocka per másodperc (fps) sebességre korlátozódik. Dolgoznak rajta, hogy a képeket *motion-jpeg* folyamként is le lehessen tölteni a kameráról, de ez a munka még nem fejeződött be. Gyakorlatban azonban 10 vagy 12fps tökéletesen elegendő a szoba felügyeletéhez.
Szükségünk lesz egy eszközre is ahol a *Motion* készítette videókat tárolhatjuk. Én általában a */var/log/vcr* könyvtárat használom a *Linux* kiszolgálómon. A használt elérési út természetesen a lemezhely lehetőségeinktől is függ. Ideális esetben egy külön fájlrendszert tudunk nyitni a *Motion* videóknak, amivel elkerülhetjük, hogy a gyökér vagy a /var fájlrendszert videó fájlokkal töltsük fel. Ezt a könyvtárat a `target_dir` változóval adhatjuk meg a *motion.conf*-ban. Következő lépésben adjuk meg a készítendő videó típusát. A *Motion 3.1.16* az *MPEG1*, *MPEG4* és *MS-MPEG4* formátumokat támogatja. Az *MPEG1* előnye, hogy egyszerű és jól támogatott formátum. Az *MPEG4* ugyanakkor szebb képet és jobb tömörítést biztosít. Az utolsó formátumot, az *MS-MPEG4*-et pedig a *Microsoft Windows Media Player* további kiegészítő bővítmények nélkül is képes értelmezni. Figyelem: az *MPEG4* és *MS-MPEG4* támogatás a *Motion 3.1.16*-os verziójában jelent meg, így nincs olyan részletesen tesztelve mint az *MPEG1* videó kezelése. Nálam azonban az *MS-MPEG4* jól bevált, és a *Windows* felhasználóknak egyszerűbb megnézni. Az *MPlayerrel* vagy más modern videó lejátszóval tetszőleges formátumú videókat nézhetünk Linux rendszeren.
A videó típusát a `ffmpeg_video_codec` változó vezérli a *motion.conf* állományban.
Ennyi alapinformációval már el tudjuk kezdeni a *Motion* használatát. Ellenőrizzük, hogy a `output_normal off`-ra legyen állítva, ugyanis különben az összes képkocka *JPEG* képe a `target_dir` könyvtárba kerül. Ez később jól jöhet a hibakeresésénél, de jelenleg csak felesleges teher.

A Motion indítása

Indítsuk el a *Motion*-t root-ként a parancssorból: */usr/local/bin/motion*. A *Motion* remélhetőleg elindul és megkezdi működését. Ha azonnal kilépne akkor valószínűleg hibát ejtettünk a beállításállományában. Hibakeresésben segít a hibaüzenet. Miután sikerült elérnünk, hogy a *Motion* elinduljon és fusson, hozzunk létre valami bemenetet. Sétáljunk a kamera előtt, vagy ami még jobb, kérjünk meg valakit, hogy tegye ezt meg. Ne feledjük el felgyújtani a lámpákat a szobában, különben a kamera nem fog túl sok mozgást észlelni. Ahogy a kamera előtt megindul a mozgás, a *Motion* elkezd kimeneti állományokat készíteni. Ha a tevékenység befejeződött, ellenőrizzük, hogy keletkeztek-e állományok a `targer_dir` könyvtárban. Nézzük meg az állományt a videólejátszónkkal. A videó döcögősnek tűnhet, hiszen csak állóképeket szedegetünk a netkameráról. A *Motion* kitölti a hiányzó képkockákat, így a videó normál sebességgel fut, a minősége pedig hozzávetőlegesen megfelel a kiskereskedésekben megfigyelhető kamerák minőségének. Ha minden jól ment ideje beállítanunk, hogy a Motion minden rendszerindításkor elinduljon.
A *Motion*-t egy `init` parancsfájl készítésével futtathatjuk minden rendszerinduláskor. A *Red Hat*-alapú rendszereken a *motion.init* állományt másoljuk a *Motion* forráskönyvtárából a */etc/init.d/motion* könyvtárba majd rootként futtassuk le a következő parancsokat:

```
# /sbin/chkconfig --add motion
# /sbin/chkconfig motion on
```

Ezek után kézzel lefuttatva próbáljuk ki, hogy a behúzófájl valóban működik-e:

```
/etc/init.d/motion start
```

Végül, aki igazán súlyos üldözési mániában szenved, az indítsa újra a rendszert és ellenőrizze, hogy a *Motion* valóban elindul és működik-e újraindítás után is.

Beállítások hangolása

Mint minden jó linuxos program a ***Motion*** is rendelkezik néhány finomhangolható képességgel. A legjobb tanács amit a ***Motion*** hangolásakor adhatunk, hogy csak egy változót változtassunk meg, majd indítsuk újra a ***Motiont*** és próbáljuk ki. Néhány beállítási változó nem egészen egyértelmű hatást gyakorolhat a többire.

Első lépésként érdemes bekapcsolni a `locate` és `text_changes` ***motion.conf*** változókat. A `locate` minden képkockán dobozt rajzol megtalált mozgás köré, a `text_changes` pedig a kép sarkában kiírja a megváltozott pixelek számát képkockánként. Ezzel a két beállítással könnyen kitalálhatjuk, hogy a kép melyik részéről gondolja a ***Motion***, hogy mozog, illetve ott mennyi mozgás van (azaz mennyi pixel változott meg a képen).

Azonnal észrevettem, hogy valószínűleg nem túl jó helyre tettem a kamerát a kiszolgálóteremben. A szobának ugyanis volt egy ablaka, amely egy másik irodahelységbe nézett. Eltartott egy ideig míg kiderítettem, hogy miért kapok annyi apró ***Motion*** filmet amikor az egyetlen mozgás a szoba apró árnyalat és megvilágítás változása volt. Végül rájöttem, hogy a másik szobában található világos színű ajtó kinyitásakor időnként fény vetül az ablakon keresztül a kiszolgálóterembe. Ez a fény verődik vissza a fém légkondicionáló fényes felületéről a kamerába. Így aztán, bár a kamera egyáltalán nem látta az ablakot, a rávetülő fény mégis hamis jelzéseket okozott. Visszatekintve, úgy kellett volna elhelyezni a kamerát, hogy ne nézzen külső fényforrások és fényes fémfelületek felé. Végül mégis inkább úgy döntöttem ott hagyom ahol van, hiszen tényleg ez volt a legjobb hely ahonnan be lehetett látni az egész szobát. A kamera mozgatása helyett, inkább a ***Motion***-t módosítottam egy kicsit.

Először is létrehoztam egy maszkállományt. Ez a kamera kimenetével azonos (tehát az ***Axis*** esetében ***640×480***) méretű, egyszerű fekete fehér kép. A fekete részeket a Motion figyelmen kívül hagyja. Az állományt a ***The GIMP*** segítségével készítettem és kifeketítettem a légkondicionáló fémrészeinek megfelelő területeket. Sajnos a ***Motion*** elég válogatós e fájl tekintetében; nyers és nem ***ASCII***, hordozható szürke skálás (***PGM***) állományként kell kimentenünk.

A ***Motion*** nem kedveli a ***The GIMP*** által létrehozott ***PGM*** állományokat. Ha egy ilyet használunk, a ***Motion*** ugyan elindul, azonban hamar kilép a következő üzenettel:

```
This is not a ppm file, starts with 'P6'
```

Néhány perces forráskód nézegetés után kiderült a hiba oka. A ***Motion*** azt szeretné, hogy a ***PGM*** állományok elején található verziószám ***P6*** helyett ***P5*** legyen. Szerkesszük át a maszk állományunkat és írjuk át az elején lévő mágikus számot ***P6***-ról ***P5***-re. Az állományt nyugodtan szerkeszthetjük `vi`-ban. A változtatás után a ***Motion*** gond nélkül beolvassa az állományt.

A módosítás csökkentette, de nem szüntette meg az üres felvételeket. Ezért egy másik módosításhoz fordultam. Megpróbáltam beállítani a ***villanykapcsoló (light switch)*** paramétert, amely a ***motion.conf*** megjegyzése szerint segíthet kiszűrni a hirtelen fényváltozásokat. Ezt teljesen hatástalannak találtam. Megpróbálkoztam a rögzítéshez szükséges megváltozott pixelek számának csökkentésével is.

A `text_changes` kimenete jól jött ilyenkor, hiszen minden egyes képkockára kiírta a megváltozott pixelek számát. Amennyiben a ***Motion*** túlságosan sok hibás filmet ment ki, megpróbálhatjuk a `text_changes` által kiírt értékek fölé emelni a határértéket.

Végül a legjobb megoldásnak a `motion_minimum_frames` érték megnövelése bizonyult. Ez azoknak a képkockáknak a számát jelenti, amelyeken változásnak kell lennie, hogy a ***Motion*** elkezdjen filmet készíteni. Ezt az értéket háromra állítva úgy találtam, hogy a fényváltozásból adódó legtöbb hamis film eltűnt. A legtöbb ilyen film ugyanis csak néhány képkocka hosszú volt, hiszen a fényváltozás egészen hirtelen történt. Ezzel szemben a valódi mozgás események általában sok képkockán keresztül tartanak. Így, ha rengeteg egy-két másodperces apró filmet találunk, javaslom emeljük meg a `motion_minimum_frames` értékét legalább háromra, esetleg még többre.

További fejlesztések

További egyelőre csak terv szinten létező, nem-program fejlesztési ötletem, hogy mozgásérzékelőt szerelünk be a szerverszoba villanykapcsolójának működtetéséhez. Ezzel ügyesen megoldanánk, hogy mindig legyen elég fény a szobában a ***Motion*** felvételeihez. Valami történik a szobában, a fény felgyullad, a ***Motion*** pedig rögzít. Mozgásérzékelő lámpakapcsolók 15 dollár körüli összegért kaphatóak a kereskedésekben és csak alapvető villanyszerelési ismereteket igényelnek. Egyelőre egyszerűen csak hagyom a ***/var/log/vcr*** tárolóhelyen felgyűlni a filmeket, aztán időnként kézzel törlöm őket. Elképzelhető, hogy lenne értelme automata módszert is kidolgozni az ilyesmi kezelésére. Jelenleg úgy gondolom, hogy a filmeket 30 naponta érdemes törölni. Nyilvánvalóan ez az egyedi igényektől függ.

A levelezőlistán mostanában jelent meg néhány kísérleti ***mjpeg*** támogatási folt. Mint korábban említettem az ***mjpeg*** azt jelenti, hogy a ***Motion*** állandó képfolyamot kér le a kameráról és nem egyesével tölti le azokat. Ezzel sokkal folyamatosabb videókat készíthetünk, bár a nettkamerákból származó mostani ***Motion*** videók amolyan igazi ***Keystone Kops*** hangulatot keltenek. A ***GNU Motion*** aktív fejlesztése tovább folyik. A levelezőlista (lásd a forrásokat) kiváló hely, ha valaki kérdéseket szeretne feltenni vagy a fejlesztésről érdeklődik. A ***Motion***-nel kapcsolatos tudásom nagy részét a levelezőlista archívumát böngészve szereztem.

Összefoglalás

A ***GNU Motion*** a számítógépipar egyik legbosszantóbb problémájára, az adattúltengés gondjaira kínál megoldást. Mi értelme a rengeteg videó-felvételnek, ha több van belőle, mint amit valaha is meg tudnánk nézni? Egy kis képvizsgálat segítségével a ***Motion*** hamar kiszűri az unalmas, változatlan videofelvételeket, melyek senkit sem érdekelnek. Az eredmény: hatékonyabb kiszolgálóterem megfigyelés és több idő amit a projektjeinkre fordíthatunk.

Linux Journal 2005. március, 131. szám

A cikk forrásai: ➲ ***www.linuxjournal.com/article/7966***

Phil Hollenback
(www.hollenback.net)

Központi, címtárszolgáltatás alapú jogosultságkezelés (2. rész)

Ki és hova jelentkezhet be? Megbízható, központi címtárral ennek szabályozása nem okozhat gondot.

A jogosultságkezelés az a folyamat, amelynek során eldöntjük, hogy X entitás, jellemzően egy személy, jogosult-e az Y erőforrás használatára. X vizsgálata meghatározása a hitelesítési folyamat feladata. A számítógépes hálózatokban a jogkezelés egyik lépése annak meghatározása és ellenőrzése, hogy az egyes felhasználók a hálózat mely számítógépeihez kaphatnak hozzáférést. Egyszerű példaként a számítógép */etc/passwd* fájljában lévő `janos:X:1234:56:/home/janos:/bin/bash` sor említhető, amely a `janos` nevű felhasználónak hozzáférést ad a helyi géphez. Ha a `janos` nevű felhasználónak több számítógép használatára szeretnénk jogot adni, akkor az összes gép */etc/passwd* fájljához hozzá kell adnunk a fenti sort.

Linux alatt jellemzően minden olyan felhasználó, aki jogot kap a helyi számítógépre való bejelentkezésre, helyi fiókkal rendelkezik. Ez arra is visszavezethető, hogy a felhasználóknak nemcsak bejelentkezési jogot kell kapniuk, de munkájuk elvégzéséhez további erőforrásokhoz, például a kezdőkönyvtárukhoz is hozzá kell férniük. Ha minden számítógépen létrehozzuk a megfelelő helyi fiókokat, akkor ez a kérdés megoldottnak tekinthető.

A helyi fiókokra épülő megoldással az a baj, hogy a fiókok egységessége nem biztosított. Azonos felhasználónévhez egy másik gépen másik felhasználóazonosító és/vagy csoportazonosító tartozhat. Ennél is nagyobb gondot jelent, ha különböző gépeken két különböző fiókhoz azonos felhasználóazonosító és csoportazonosító tartozik. Lehetséges például, hogy a `janos` nevű felhasználó az 1-es számítógépen az 1234 felhasználóazonosítót és az 56 csoportazonosítót kapja, míg a `julia` nevű felhasználóhoz a 2-es számítógépen ugyancsak az 1234 felhasználó- és az 56 csoportazonosító tartozik. Ez a felállás megosztott erőforrások használatakor komoly biztonsági kockázatot jelent. Például egy *NFS*-kiszolgáló számára a két fiók azonosnak látszik, tehát a két felhasználó akadálytalanul törölheti egymás fájljait.

A egységesség problémáján úgy lehetünk úrrá, hogy minden szükséges információt egyetlen központi hitelesítési adatforrásból veszünk, természetesen biztosítva a számítógépeknek a hozzáférést ehhez a forráshoz. A címtárszolgáltatások pontosan ezt a célt szolgálják. Napjainkban a két legelterjedtebb, központi hitelesítésre használt címtárszolgáltatás a ***NIS (hálózati információs szolgáltatás, network information service,*** korábban ***yellow pages,*** azaz ***sárga oldalak,*** röviden ***YP*** névvel futott***)*** és az ***LDAP (lightweight directory access protocol, egyszerű címtárelérési protokoll)***.

A NIS és az LDAP összehasonlítása

Amikor arra a kerül a sor, hogy ki kell választanunk, címtárszolgáltatásként a *NIS*-t vagy az *LDAP*-t használjuk, nem árt néhány tényezőt figyelembe vennünk. Ha cégünk már rendelkezik *LDAP*-kiszolgálóval, akkor egyszerűnek tűnik a megoldás: bővíteni kell a hitelesítési és jogkezelési adatokkal. Csakhogy a vállalati *LDAP*-kiszolgálókat általában névjegyek tárolására és hasonló, valóban egyszerű feladatokra használják. A jogkezelési adatok hozzáadása komoly terhelést róna a kiszolgálóra, hiszen a programok által indított, felhasználónév, felhasználóazonosító, csoportazonosító stb. lekérdezésére irányuló kéréseket mind meg kell válaszolnia. Általában érdemesebb egy további, kizárólag a jogkezeléssel foglalkozó *LDAP*-kiszolgálót üzembe állítani. A címtár felé irányuló lekérdezések sokszínűsége miatt a teljesítmény hangolása is rendkívül nehéz. A gyakoribb lekérdezéseket úgy gyorsíthatjuk fel, hogy az összes szükséges *LDAP* indexmegadást hozzáadjuk a slapd.conf fájlhoz, ám túlságosan sok indexmegadást sem célszerű hozzáadni, mert ezzel az *LDAP* adatbázisfájljainak növekedését okozzuk, ami miatt viszont újfent csökkenni fog a sebesség.

Az *LDAP* azokban a hálózatokban nyújt jobb megoldást, ahol sok *UDP*-csomag vész el, ugyanis *TCP/IP* alapú, vagyis az újraküldések elvégzését a hálózati rétegre bízza. A *NIS* ezzel szemben *UDP* feletti ***távoli eljáráshívást (remote procedure call, RPC***) alkalmaz. Esetében minden eldobott csomag egy válasz nélkül maradt *NIS*-lekérdezést jelent, amelyet az ügyfélnek meg kell ismételnie. Azt, hogy hálózatunkra mennyire jellemző a csomagvesztés, a `netstat -s -u` parancsot különböző gépeken, különböző időpontokban kiadva vizsgálhatjuk meg. Jó esetben a parancs csak elenyésző számú hibát jelez.

Írásomban elsősorban a *NIS*-sel foglalkozom, mivel üzembe helyezése rendkívül egyszerű, és ha elégedetlenek vagyunk vele, akkor rendkívül könnyen áttérhetünk róla az *LDAP* használatára. A *PADL Software Pty, Ltd.* több nyílt forrású, a *NIS* adatfájlok *LDAP* fájlokká alakítására alkalmas programot is készített. (Lásd az internetes forrásokat.) A teljesítményhangolás természetesen mindenképpen ránk marad. Ha *LDAP*-ról akarunk *NIS* használatára áttérni, akkor az átalakítást magunknak kell megoldanunk.

A NIS-kiszolgálók beállítása

A *NIS*-kiszolgálók nem igényelnek komolyabb hardveres erőforrásokat. Gyakorlatilag bármilyen átlagos gépen futtathatók. Ugyanakkor a szolgáltatást érdemes lehet külön gépre telepíteni. Nálunk, a *Stanford Linear Accelerator Centerben (SLAC)* 500 *Linuxot* és *Solarist* futtató ügyfélgépet szolgálunk ki egyetlen régi *Sun Netra T1* géppel. Négy *NIS*-kiszolgáló szolgálja ki 700 *Solaris* és *Linux* alapú asztali számítógépünket, és további hat a körülbelül 2500 darab, szintén Solaris és *Linux* alapú számítási kiszolgálót. Az ügyfelek nem teljesen egyenletesen oszlanak el a kiszolgálók között.

A mester kiszolgáló beállításai

Jelentkezzünk be arra a gépre, amelyre telepíteni szeretnénk a mester *NIS*-kiszolgálót, majd ellenőrizzük, hogy a legújabb `portmap`, `ypserv` és `yp-tools` *RPM*-ek telepítve vannak-e. Ha nem, töltsük le és telepítsük őket. Az alábbi parancsokat kivétel nélkül rootként kell kiadni. Indítsuk el a `portmapper` démont:

```
# service portmap start
```

A következő lépés új *NIS*-tartományunk nevének megadása. A név tetszőleges, de nyilván érdemes olyat választani, ami tükrözi cégen belüli szervezetünk, részlegünk nevét. Ha például *nis.pelda.com* a teljes *pelda.com NIS*-tartománya, akkor a tervezési részleghez választhatjuk a *terv.pelda.com*-ot.
A mester kiszolgálón a *NIS*-tartománynevet a következő paranccsal állíthatjuk be:

```
# domainname nis.pelda.com
```

A következő sort:

```
NISDOMAIN=nis.pelda.com
```

pedig a */etc/sysconfig/network* fájlhoz kell hozzáírnunk.

A *NIS*-kiszolgáló elérhetőségét egy */var/yp/securenets* nevű, alábbi tartalmú fájl létrehozásával korlátozhatjuk:

```
# netmask        # network
255.255.255.0    192.168.0.0
```

Ez egy fontos biztonsági lépés, ne feledkezzünk meg róla! Ha ezt a fájlt nem hozzuk létre, bárki képes lesz lekérdezéseket intézni *NIS*-kiszolgálónkhoz.

Haladjunk tovább: meg kell határoznunk, milyen adatokat akarunk *NIS* alatt tárolni. Jogosultságkezelési célokra a */etc/group* és a */etc/passwd* fájl, továbbá egy `netgroup` nevű dolog elegendő lesz. Ennél azonban sokkal többre is van lehetőség. Ötleteket a *NIS*-kiszolgáló */var/yp/Makefile* fájljából lophatunk.
Az alábbiakban bemutatom, hogyan történik az említett három fájl *NIS* segítségével végzett terjesztésének beállítása. Módosítsuk a *NIS*-térkép adatbázisfájljait létrehozó *Makefile*-t:

```
# cp /var/yp/Makefile /var/yp/Makefile.save
# vi /var/yp/Makefile
```

Az alábbi két beállítás értékét `true`-ról `false`-ra változtatva megakadályozhatjuk a *passwd* és a *shadow*, illetve a *group* és a *gshadow* fájlok egyesítését:

```
MERGE_PASSWD=false
MERGE_GROUP=false
```

Módosítsuk a *NIS* adatforrásait tartalmazó könyvtárak nevét:

```
YPSRCDIR = /etc/NIS
YPPWDDIR = /etc/NIS
```

Azokat a fájlokat, amelyekből nem akarunk *NIS* adatbázisokat készíteni, tegyük megjegyzésbe. Én csak az alábbi hármat hagytam meg:

```
GROUP        = $(YPPWDDIR)/group
PASSWD       = $(YPPWDDIR)/passwd
NETGROUP     = $(YPSRCDIR)/netgroup
```

Az `all:` kezdetű, az összes lehetséges *NIS*-térkép listáját tartalmazó bejegyzést tegyük megjegyzésbe. Adjuk hozzá az alábbi új sort:

```
all:    passwd group netgroup
```

Ügyeljünk a tabulátorokra! A *Makefile* fájlokban a parancsok igazítására kizárólag tabulátorokat használjunk, szóközöket ne.
Hozzuk létre a *Makefile*-ban megadott adatforrás könyvtárat:

```
# mkdir /etc/NIS/
# chmod 700 /etc/NIS
```

majd helyezzünk bele egy passwd fájlt:

```
# grep -v '^root' /etc/passwd > /etc/NIS/passwd
```

A fájlból vegyük ki a root fiókot, valamint az összes egyéb rendszerfiókot; kizárólag a valódi felhasználói fiókokat hagyjuk meg benne.
Ha a */etc/passwd*-t még mindig használjuk titkosított jelszavakkal, akkor itt az ideje, hogy az előző cikkben (*Linuxvilág, 2005. március*) ismertetett módon áttelepít-

sük őket *Kerberos 5* alá. Ha ezt nem tesszük meg, akkor a titkosított jelszavak hozzáférhetőkké válnak, amikor a *passwd* fájlt továbbadjuk a szolga *NIS*-kiszolgálók vagy a *NIS*-ügyfelek felé.
Gyűjtsük össze a helyi */etc/passwd* fájlokat az összes olyan gépről, mely az új *NIS*-tartomány része lesz. Távolítsuk el belőlük a rendszerfiókokat, majd az összes fájlt másoljuk egybe:

```
% cat jelszo_1 jelszo_2 jelszo_3 ... >
➥osszegyujtott_jelszavak
```

Az alábbi paranccsal távolítsuk el a kettős bejegyzéseket:

```
% sort osszegyujtott_jelszavak | uniq >
➥egyedi_jelszavak
```

Ellenőrizzük a fennmaradt bejegyzések egységességét:

```
% cut -d':' -f1 egyedi_jelszavak | sort | uniq -c
➥| \
egrep -v "\s*1"
```

Ha a parancs bármit is ad kimenetként, akkor van két különböző, de azonos fióknévvel rendelkező bejegyzésünk. Ha az eltérés nem az *UID* vagy a *GID* mezőben jelentkezik, akkor egyszerűen válasszuk ki az egyik bejegyzést, és töröljük a másikat. Ha az *UID* vagy a *GID* mező tér el, akkor fel kell oldanunk az ütközést, ami akár egészen bonyolult feladat is lehet. Végezzünk újabb ellenőrzést; van-e két azonos *UID*-vel rendelkező fiók?
Ha a:

```
% cut -d':' -f3 egyedi_jelszavak | sort | uniq -c
➥| \
egrep -v "\s*1"
```

parancs bármilyen kimenetet is ad, akkor igen. A második kapott szám a kettős *UID*. Az ütközés feloldása ebben az esetben is fárasztó feladat lehet. Hasonló módon az összes */etc/group* fájlt is egyesítenünk kell.
A kapott fájlokat másoljuk a */etc/NIS/passwd* és a */etc/NIS/group* elérési út alá. A netgroup fájlt egyelőre hagyjuk ki, majd később foglalkozunk vele.
Indítsuk el a mester *NIS*-kiszolgálót:

```
# service ypserv start
```

A *NIS*-térképeket az alábbi paranccsal:

```
# /usr/lib/yp/ypinit -m
```

illetve a megjelenő utasításokat követve vehetjük használatba.
Ha a mester *NIS*-kiszolgáló számára az összes *NIS*-térképet elérhetővé akarjuk tenni, akkor ezt a gépet *NIS*-ügyfélnek is be kell állítanunk. Ellenőrizzük, hogy ez a *NIS*-ügyfél csak a mester *NIS*-kiszolgálóhoz tud-e kötni, ezzel megelőzhetjük, hogy például egy áramkimaradás után az induló gépek között körkörös függés alakuljon ki.

A szolga kiszolgáló beállítása

A szolga *NIS*-kiszolgálók olyan *NIS*-ügyfelek, melyek a mester *NIS*-kiszolgálótól kapott térképeket továbbterjesztik a többi *NIS*-ügyfél felé. Ellenőrizzük, hogy a legújabb `portmap`, `ypserv`, `ypbind` és `yp-tools` *RPM*-ek az összes szolga kiszolgáló gépre telepítve vannak-e. Egy szolga *NIS*kiszolgáló üzembe helyezésének első lépése az, hogy *NIS*-ügyfélként állítjuk be. Ennek módjáról a következő részben lesz szó.

Ha a *NIS*-ügyfél beállítása megtörtént, indítsuk el:

```
# service ypbind start
```

A mester *NIS*-kiszolgálón adjuk hozzá az új szolga *NIS*-kiszolgáló nevét a */var/yp/ypservers* fájlhoz, majd futtassuk le az alábbi parancsokat:

```
# cd /var/yp
# /usr/lib/yp/makedbm ypservers
/var/yp/nis.example.com/ypservers
```

A mester *NIS*-kiszolgálón a */etc/YP/Makefile* fájlban a `NOPUSH` megadást is meg kell változtatnunk, true értékről false-ra, a frissített *NIS*-térképek ugyanis csak ekkor kerülnek át a mester kiszolgálóról a szolgára vagy szolgákra.
Visszatérve az új szolga *NIS*-kiszolgálóra, elindításához a következő parancsot kell kiadnunk:

```
# /usr/lib/yp/ypinit -s nismester
```

Itt a `nismester` a mester *NIS*-kiszolgáló neve. Ennek egy teljesen minősített tartománynévnek kel lennie, feltéve, hogy *DNS*-kiszolgálónk a névkeresésekre ilyet ad vissza.
A mester *NIS*-kiszolgálóról másoljuk a */var/yp/securenets* fájlt az új szolga kiszolgálóra, majd az alábbi paranccsal indítsuk el az új szolgát:

```
# service ypserv start
```

A katasztrófa utáni helyreállítás tervét ne felejtsük el frissíteni, és jelezzük benne, hogy a szolga *NIS*-kiszolgáló függ a mester *NIS*-kiszolgálótól.

Az ügyfelek beállítása

Az összes ügyfélre telepítsük a legújabb `ypbind`, `yp-tools` és `portmap` *RPM*-eket. A */etc/yp.conf* fájlt írjuk át; adjuk meg benne a *NIS*-kiszolgálót:

```
ypserver nismester.pelda.com
```

A szolga kiszolgálókat külön-külön sorokban kell megadnunk. Az ügyfeleken próbáljuk véletlenszerűen felsorolni a kiszolgálókat, így a terhelést viszonylag egyenletesen tudjuk elosztani közöttük.
A */etc/sysconfig/network* fájlhoz az alábbi sort írva adjuk meg az ügyfél *NIS*-tartományát:

```
NISDOMAIN=nis.pelda.com
```

Állítsuk be a *NIS*-tartománynevet:

```
# domainname nis.pelda.com
```

Indítsuk el a portmappert:

```
# service portmap start
```

illetve a *NIS*-ügyfelet:

```
# service ypbind start
```

az összes ügyfélen.
Az `ypwhich` parancsnak most azt a *NIS*-kiszolgálót kell kiírnia, amelyhez az ügyfél kötődik.
A NIS-térképek tartalmát az `ypcat` paranccsal tudjuk megvizsgálni. Például:

```
% ypcat passwd
```

Következő teendőnk az, hogy az ügyfeleket úgy állítsuk be, hogy minden keresést *NIS* segítségével végezzenek. Ezt a névszolgáltatás-kapcsoló */etc/nsswitch.conf* beállító fájljának módosításával érhetjük el. A `passwd`, a `group` és a `netgroup` bejegyzéseket a következőképpen kell módosítanunk:

```
passwd:       compat
group:        files nis
netgroup:     nis
```

A fentiek értelmében a *csoport (group)* kereséseket a helyi */etc/group* fájllal kell kezdeni, majd egy *NIS*-kereséssel kell folytatni. A hálózati *csoportokat (netgroup)* kizárólag *NIS* alatt kell keresni. A `passwd` mögött álló `compat` kulcsszóról később ejtünk szót.
Megjegyezném, hogy az *nscd névszolgáltatás-gyorsítótárazó (name service caching)* démonnak időnként gondjai vannak belső gyorsítótárának frissítésével. Ennek hatásaként előfordulhat, hogy a valamelyik *NIS*-térképben végrehajtott módosítások egy-egy ügyfélen nem látszanak. Ilyenkor az egyetlen megoldás az adott gép nscd-jének újraindítása.

Jellemző felhasználások

Ha *NIS* alól információkat akarunk lekérdezni, akkor két paranccsal mindenképpen érdemes megismerkednünk; az egyik az `ypcat`, a másik az `ypmatch`. Az `ypcat` végigmegy az adott *NIS*-térkép összes kulcsán, és kiírja a benne szereplő értéket. Például az `ypcat passwd` paranccsal a passwd NIS-térkép bejegyzéseit listázhatjuk ki. Az `ypmatch` a megadott *NIS*-térképből egy vagy több kulcs értékét írja ki; az `ypmatch julia passwd` parancs például a julia nevű fiók `passwd` bejegyzését adja meg.

NIS csoporttérkép

A *NIS* csoporttérkép jellegzetes használata a több felhasználó közötti fájlmegosztás lehetővé tétele. A megoldás helyi és *NFS*-en található fájlokkal egyaránt működik. Lássuk a beállításokat. Tegyük fel, van két felhasználónk (az eljárás tetszőleges számú felhasználóval is működik), ők a következő bejegyzésekkel rendelkeznek a `passwd` térképben:

```
julia:*:1234:42:Julia:/home/julia:/bin/bash
janos:*:5678:57:Janos:/home/janos:/bin/bash
```

A fentiek értelmében az ***elsődleges*** csoportazonosító `julia` esetében 42, `janos`-nál pedig 57.
A *NIS* csoporttérképpel egy további, ***másodlagos*** csoporttagságot is megadhatunk a fiókokhoz. A

```
tervezetX:*:127:julia,janos
```

csoportbejegyzés egy új, `tervezetX` nevű csoportot ad meg, jelszó nélkül (*), 127-es csoportazonosítóval és két taggal.
A csoportfájlban nem lehetnek megjegyzések.
Ha most egy könyvtárra olvasási/írási/futtatási jogot adunk a `tervezetX` csoportnak:

```
# mkdir /tervezetek/X/
# chgrp tervezetX /tervezetek/X/
# chmod g+wrx /tervezetek/X/
```

akkor a `tervezetX` csoport minden tagja olvasási/írási/futtatási jogot kap a könyvtár fájljaira. Lehetséges, hogy a felhasználónak először ki kell adnia a `newgrp tervezetX` parancsot.
Ha hozzá kell adnunk egy fiókot egy csoporttérképhez, illetve, ha el kell távolítanunk belőle ilyet, akkor azt a mester *NIS*-kiszolgálón, a */etc/NIS/group* fájl módosításával és a következő parancsok kiadásával tegyük meg:

```
% cd /var/yp
% sudo make group
```

Ekkor létrejön az új csoporttérkép, amelynek révén az összes ügyfél azonnal láthatja a változásokat. A módosítás elvégzéséhez az ügyfelek közelébe sem kell menni – hiszen minden központosítva van egy helyre, a *NIS*-kiszolgálóra.

NIS hálózati csoportok

A ***hálózati csoportok (netgroup)*** teljesen eltérők a csoportoktól. A hálózati csoportok két típusra oszthatók, felhasználói és állomás hálózati csoportokra. Mindkét típusú hálózati csoport további hálózati csoportokat is tartalmazhat tagként, vagyis a hálózati csoportok hierarchiába rendezhetők. Mindkét típus megadása ugyanabban a ***netgroup*** fájlban történik. Ebben a fájlban megjegyzések is lehetnek.
A */etc/NIS/netgroup* fájl állomás hálózati csoport megadásai így néznek ki:

```
# Tervezetcsoportok csoportja:
tervezetek \
       tervezetA \
       tervezetB \
       tervezetX
# Az X tervezet állomáscsoportja
tervezetX \
         (allomas1.pelda.com,-,) \
         (allomas2.pelda.com,-,) \
         (allomas3.pelda.com,-,)
```

A fenti állomás hálózati csoportok révén lehetővé válik, hogy például egy *NFS* tárterületet csak a munkaállomások egy részéről lehessen elérni. *NFS* kiszolgálónk */etc/exports* fájljában például a következő elérhetőségeket írhatjuk elő:

```
# a /tervezetek könyvtár elérhetővé tétele az
# összes olyan gép számára,
# amely a "tervezetek" hálózati csoportban
# található
/tervezetek     @tervezetek(rw,root_squash)
# a 'tervezet X' csak azokról a gépekről
# érhető el, amelyek a "tervezetX"
# hálózati csoport
# tagjai
/tervezetek/X           @tervezetX(rw,root_squash)
```

Az állomások hozzáadása a hálózati csoportokhoz, illetve kivétele belőlük az előbbihez hasonló módon a mester *NIS*-kiszolgálón található */etc/NIS/netgroup* fájl átírásával végezhető el. A *NIS*-térképet a `cd /var/yp; sudo make netgroup` parancsokkal frissíthetjük. A változások mindenhol azonnal megjelennek.

Felhasználói hálózati csoportok

A felhasználói hálózati csoportoknak felhasználók a tagjai, szerepük elsősorban az egyes számítógépekre való bejelentkezések korlátozására terjed ki. A felhasználói hálózati csoportok megadása kicsit eltér az állomás hálózati csoportokétól:

```
# Tervezet felhasználói csoportok csoportja
u-tervezetek \
        u-tervezetA \
        u-tervezetB \
        u-tervezetX

# Az X tervezet felhasználócsoportja
u-tervezetX \
        (-,julia,) \
        (-,janos,) \
        (-,norbert,)
```

A ***felhasználói (user)*** hálózati csoportokat hagyományosan az `u-` előtaggal különböztetik meg az állomás hálózati csoportoktól.
A fenti definíciók birtokában a gépek helyi */etc/passwd* fájljában lévő, hasonló típusú bejegyzésekkel tudjuk engedélyezni vagy éppen tiltani a bejelentkezést az egyes gépekre. A *passwd* fájlok legvégéről vegyük ki a + jelet, ha van ilyen:

- A hozzáférés az u-tervezetek hálózati csoportban lévő fiókokra történő korlátozása:

```
+@u-tervezetek
```

- A hozzáférés korlátozása az `u-tervezetX` hálózati csoport tagjaira:

```
+@u-tervezetX
```

- Hozzáférés biztosítása azoknak, akik az u-tervezetek csoportnak tagjai, de az `u-tervezetX` csoportnak nem:

```
-@u-tervezetX
+@u-tervezetek
```

Ez esetben fontos a sorrend. Mindig az első egyezés szabja meg, hogy mi történik.

- Engedély megadása az `u-tervezetA` csoport tagjainak és a `norbert` fióknak:

```
+@u-tervezetA
+norbert
```

A `norbert` fiókkal kapcsolatos adatok (kezdőkönyvtár, bejelentkezési héj stb.) a *NIS passwd* térképből származnak. Kifejezett fiókneveket ide inkább ne helyezzünk, mert ezeknek a bejegyzéseknek a kezelése nincs központosítva.
A +/- jelekre alapuló írásmód működéséhez az ügyfelek */etc/nsswitch.conf* fájljába a következő sort kell beilleszteni:

```
passwd:         compat
```

Összefoglalás

Ha a *NIS*-kiszolgáló telepítésének és a jogosultság-kezelési adatok egységesítésének gondján-baján túlestünk, a központosítással könnyebbé válik az életünk. A hálózati csoportok révén összetett, kifinomult, központi hozzáférésvezérlést valósíthatunk meg.

Linux Journal 2005. március, 131. szám

Dr. Alf Wachsmann
1999 óta a Stanford Linear Accelerator Center (SLAC) munkatársa. Ő felelős az önműködő Linux-telepítések minden mozzanatáért, egyaránt ide értve a farmok csomópontjainak, a kiszolgálóknak és az asztali gépeknek a kezelését. Munkája során elsősorban az aktív fájlkészletek (AFS) támogatásával, a Kerberos 5-re való áttéréssel, egy felhasználónyilvántartó tervezettel és felhasználói tanácsadással foglalkozik.

KAPCSOLÓDÓ CÍMEK

- www.tldp.org/HOWTO/NIS-HOWTO/index.html
- www.padl.com/OSS/MigrationTools.html

Az OSCAR és a bioinformatika

Az új fürt-telepítő és -irányító eszközökkel egy óra alatt létrehozhatunk és a kutatómunkánk szolgálatába állíthatunk egy 64 csomópontos számítógépfürtöt.

Az *OSCAR (Open Source Cluster Applications Resources)* nyílt forrású fürtöző programkészlet) projekt körülbelül négy éve indult. Az elv 2000 januárjában került először előterjesztésre, az első szervezeti találkozóra pedig ugyanez év áprilisában került sor. A csoport felismerte, hogy a fürtök összeállítása meglehetősen időrabló és rutinszerűen ismétlődő tevékenység, ezért a projekt céljául ennek a folyamatnak az önműködővé tételét tűztük ki. A csoport azt remélte, hogy ennek hatására szélesebb körben terjed majd el a fürtök használata, és alkalmassá válik a tudományos életben és a magánszektorban való alkalmazásra is.

Az *OSCAR* projektet a nyílt tagsággal rendelkező nem hivatalos testület, az *OCG (Open Cluster Group)* tanácsadó csoport felügyeli. Az *OCG* azért küzd, hogy növelje a fürtözés alkalmazhatóságát a *nagyteljesítményű számítások (HPC, high-performance computing)* kutatásának és fejlesztésének területén. A csoport – az *OSCAR projekthez* hasonlóan – a tudományos élet és az ipar jeles képviselőinek az irányítása alatt áll. Legfontosabb tagjai közt találjuk a következő intézményeket: *Bald Guy Software*, *BC Genome Tudományos Központ*, *Dell*, *Indiana Egyetem*, *Intel*, *Louisiana Műszaki Egyetem*, *Oak Ridge Nemzeti Labotartórium*, *Revolution Linux* és a *Sherbrooke Egyetem*.

Az *OSCAR*-csoport az *OCG* egyik munkacsoportja. Emellett további projektek is működnek, így a *HA-OSCAR (high-availability, magas rendelkezésre állású)*, *Thin-OSCAR (lemez nélküli)* és *SSS-OSCAR (skálázható)*. Az *OCG*-ről további információkat a cikkhez tartozó, a világhálón elérhető hivatkozások közt találhatunk.

Az *OSCAR* legelőször 2001 áprilisában jelent meg, azóta két fő változatot bocsátottunk ki. A kibocsátási ciklus rendszerint egybeesik az évente novemberben megrendezésre kerülő *SuperComputing* konferenciával. Az írás idején aktuális változat a 3.0, amit a *SuperComputing04*-re tervezett 4.0 változat kibocsátása követne.

Az *OSCAR* célja, hogy a *HPC*-fürtök telepítésének, programozásának és karbantartásának lehető legjobb eszközét nyújtsa a felhasználók számára. Sok olyan nyílt forrású összetevőt találunk, amely jól működik egy *HPC*-környezetben, de ehhez szükség van a megfelelő beállítások elvégzésére. Az *OSCAR* jelenti az összekötő anyagot ezeknek az összetevőknek egy jól működő eszközkészletté történő egyesítéséhez. A projekt a közepes méretű (50 és afeletti csomópontokból álló) fürtöket célozza meg, mivel a visszajelzések alapján a manapság használt fürtök többsége ebbe a méretkategóriába esik.

A *OSCAR* a következő összetevőkből épül fel:

- *Rendszerfelügyelet: System Installation Suite (SIS, rendszertelepítő csomag)*, *Cluster Command and Control (C3, fürtvezérlő parancsfelület)* és az *OPIUM (felhasználókezelő)*.
- *HPC-eszközök*: párhuzamos programozói könyvtárak: *MPICH*, *LAM/MPI* és a *PVM*; kötegelő rendszerek: *OpenPBS/MAUI*, *Torque* és az *SGE*; megfigyelő eszközök: *Ganglia* és *Clumon*; és egyéb külső fejlesztésű *OSCAR*-csomagok.
- *Alapinfrastruktúra/irányítás: OSCAR Database (ODA, OSCAR adatbázis)* és az *OSCAR Package Downloader (OPD, OSCAR csomagletöltő)*.

Az *OSCAR* fejlesztői egymástól távol dolgoznak, az aktuális fejlesztési kérdéseket heti telefonkonferenciákon vitatják meg. A csoport ezen kívül évente rendez olyan találkozókat, ahol a jövőbeli változatokba kerülő új tulajdonságokat találják ki és beszélik meg. Évente egy tudományos tanácskozás is megrendezésre kerül, rendszerint a *HPCS* (a nagyteljesítményű számítási rendszerek és alkalmazások nemzetközi tudományos tanácskozása) keretein belül, amelyen a felhasználók adhatják elő az *OSCAR*-ral kapcsolatos tapasztalataikat és a *HPC*-vel kapcsolatos fejlesztési eredményeiket. A második *OSCAR Symposium* 2004 májusában, a kanadai *Winnipegben* került megrendezésre, az itt készült anyagok bárki számára hozzáférhetőek.

Bevezetés a bioinformatikába

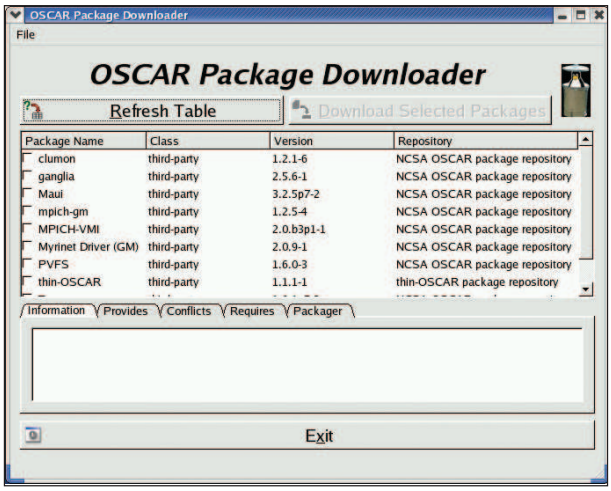
A bioinformatika a biológia és a számítástudomány egyesüléséből keletkezett, gyorsan fejlődő tudományterület, amely nagy lehetőségeket hordoz a *HPC* területe számára. Egyszerűen megfogalmazva a bioinformatika az olyan biológiai adatok számítógépes eljárásokkal és rendszerekkel történő feldolgozásával foglalkozik, mint a *DNS*, *RNS*, a fehérjék és egyéb szabályozó alkotóelemek.

1. kép Az OSCAR telepítésének főmenüje. Kevesebb, mint tíz lépés, és a fürt készen áll a számítások megkezdésére!

A biológiai adatok rendszerint karaktersorozatok, amelyek elemzése rendszerint karakterlánc-műveleteket jelent, ezért a legtöbb bioinformatikus munkaeszközéül a *Perl* nyelvet választotta. Sok nyílt forrású *Perl* programot író programozó járul hozzá a *Bioperl* kifejlesztéséhez, amely a bioinformatikai elemzésekre specializálódott *Perl*-modulok gyűjteménye. A *Java* nyelvet nagyobb, főleg grafikai felületet is igénylő projektekhez használják. Megfigyelhető még a területen a *Python* térnyerése is, ahogy egyre több programozó fedezi fel ennek a viszonylag új, de nagyon hatékony nyelvnek a könnyű használhatóságát és jó olvashatóságát.
A bioinformatikus közösségben nagyon elterjedt a *Linux* fürtök használata, mivel az elemzések egyre hosszabb ideig futnak és gyakran ismétlődő feladatokat jelentenek.
A *Linux* fürtök ideális eszközt jelentenek az ilyen egymástól függetlenül futtatható, párhuzamos feladatok végrehajtására. Ezek nem tekinthetők igazi párhuzamos programoknak, mivel nincs szükségük az *MPI*-hoz hasonló párhuzamos programozói könyvtárakra. A bioinformatika elterjedt eszközét az olyan fürtök jelentik, melyek több parancsfájlt és algoritmus futtatnak különböző bemenetekkel, külön címzési tartománnyal rendelkező processzorokon.

Egy jellemző OSCAR-telepítés áttekintése

Egy fürt telepítése az *OSCAR* eszközkészlettel egyszerű folyamat, ha telepítettünk korábban *Linuxot*, nem sok gondunk adódhat.

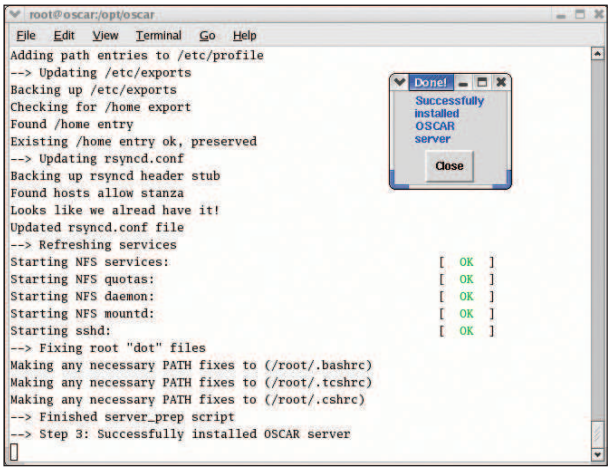


2. kép Az OSCAR Package Downloader (csomagletöltő) – ebben a menüpontban van lehetőségünk további csomagok letöltésére

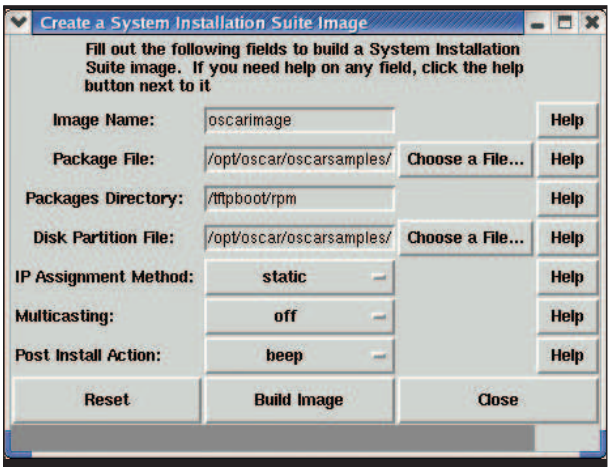
Jelenleg az *OSCAR* három hivatalosan támogatott *Linux* rendszercsomagra telepíthető: *Red Hat 8.0*, *Red Hat 9.0* és *Mandrake 9.0*. A *Linux* telepítésnek tartalmaznia kell valamilyen *X* ablakrendszert, amilyen a *KDE* vagy a *GNOME*, ettől eltekintve egy tipikus, programfejlesztő eszközökkel telepített munkaállomás beállításainak megfelelőnek kell lennie.
Miután feltelepítettük és beállítottuk a *Linuxot* a központi csomópont gépén, letölthetjük az *OSCAR* tar-csomagját a projekt honlapjáról és a kicsomagolás után elvégezhetjük a beállító, `make` és `make install` lépéseket.
Alapértelmezésben az *OSCAR* telepítése az */opt/oscar* könyvtárba történik, de ezt a *configure* parancs `–prefix` beállításával megváltoztathatjuk. Az *OSCAR* telepítését követően elindíthatjuk az *OSCAR* varázslót, amely lépésről lépésre végigvezet a fürtünk beállításán.
A varázsló indításához lépjünk be az */opt/oscar* könyvtárba majd gépeljük be az

```
./install_cluster ethX
```

parancsot, ahol az `ethX` a fürt hálózatának felületét jelenti.
Az *OSCAR* számos előre összeállított csomaggal érkezik. A más gyűjteményből származó csomagok is érdeklődésre tarthatnak számot. Ezek letöltéséhez kattintsunk az *OSCAR* varázsló ***Download Additional OSCAR Packages (további OSCAR-csomagok letöltése)*** feliratú gombra és válasszuk ki a csomago(ka)t.
Ezután kiválaszthatjuk a telepíteni kívánt OSCAR-csomagokat. A csomagok három fő csoportba sorolhatók: központi csomagok, tartozék-csomagok és külső fejlesztésű csomagok. A központi csomagokat mindenképpen telepítenünk kell, ezek kijelölését nem tudjuk törölni. A tartozék-csomagok azok, amelyeket az OSCAR fejlesztőcsapata telepítésre javasol, a maradékot pedig a külső forrásokból származó csomagok alkotják.
A csomagok beállításait a ***Configure Selected OSCAR Packages (a kiválasztott OSCAR-csomagok beállításai)*** menüben változtathatjuk meg.



3. kép Egy beavatkozást nem igénylő lépés: a kiszolgáló csomagjainak telepítése



4. kép Ügyfél-képfájl létrehozása a felhasználó által megadott csomaglista és partíciós tábla alapján

A következő lépés az ***OSCAR*** kiszolgáló csomagjainak (***Server Packages***) telepítése. Ez alapvetően a kiszolgálón használt csomagok telepítésére szolgál, és nem igényel felhasználói beavatkozást. A telepítés befejezéséről egy ablak megjelenése tájékoztat.

Most következik az érdekes rész. A ***Build OSCAR Client Image*** lépéssel létrehozhatunk egy ügyfél-képfájlt. Itt beállíthatunk néhány lehetőséget a létrehozandó képfájl számára, majd feltölthetjük a képfájlt az ügyfélcsomópontokra. Megadhatjuk az alapképfájl ***RPM***-csomagjainka listáját, eldönthetjük, hogyan legyen a merevlemez partícionálva, és elvégezhetjük az IP-cím hozzárendelését. Végül kiválaszthatjuk a képfájl áthelyezését követő műveletet, például a gép újraindítását.

A ***Define OSCAR Clients (az OSCAR-ügyfelek megadása)*** lépésben megadhatjuk a tartománynevet, az ügyfél nevét, a munkafolyamathoz használandó csomópontok számát és egyéb hálózati beállításokat. Az ***Add clients*** feliratú gombra kattintva érvényesítjük a beállításokat és már majdnem kész is a fürt.

A következő lépés a fürt hálózatának üzembe helyezése. Itt indíthatjuk el az ügyfélcsomópontokon a rendszert a ***PXE*** vagy floppy segítségével, majd az ***OSCAR*** központi csomópontja összegyűjti a ***MAC***-címeket, amiket a konkrét gépekhez rendelhetünk. Ezután rögtön megtörténik az ügyfélgépek képfájljainak üzembe helyezése. A merevlemez sebességétől függően ez gépenként 10-30 percet vehet igénybe. Egy fürt üzembe helyezésekor egyszerre több csomóponton is folyhat ez a művelet. Mi rendszerint 10 gépen indítjuk el párhuzamosan, hogy elkerüljük a központi csomópont túlterhelését. Ezzel a fokozatos eljárással egy 64-csomópontos fürt üzembe helyezése nem tarthat tovább egy óránál.

A csomópontok képfájljainak felhelyezése és a gépek újraindítása után folytathatjuk a következő lépéssel, amely A ***fürt beállításainak befejezése (Complete the Cluster Setup)***. Ez megint nem igényel felhasználói beavatkozást, ekkor hajtódnak végre a végső telepítési műveletek és egyéb tisztító folyamatok.

Végül nem árt ellenőrizni a fürt beállításait a ***Test Cluster Setup*** lépéssel. Ez egy tesztsorozatot futtat le a fürt telepítésének és a független csomagok ellenőrzésére. Ha minden jól megy, akkor a tesztek sikeresek lesznek, az üzembe helyezés befejeződött, a fürt készen áll a számítási feladatok végrehajtására.

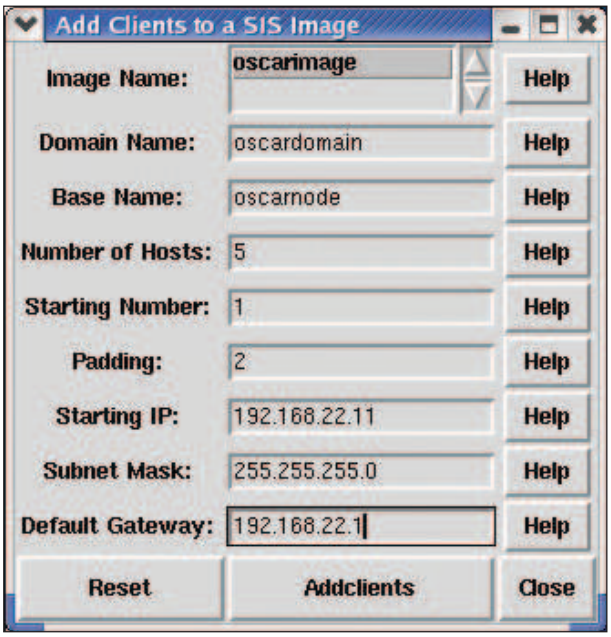
Az ***OSCAR*** eszközkészlet telepítése egyszerű és általában minden hardveren működik. Ha mégis valamilyen problémával találkoznánk, a levelezőlistákon kérhetünk segítséget. A kérdéseket első körben az ***oscar-users*** listához intézhetjük. A központi csapat nagy része rendszeresen olvassa a listát és a többi felhasználó is a segítségünkre lehet. Ha a fejlesztéssel kapcsolatos kérdés merülne fel, erre az esetre az ***oscar-devel*** lista áll rendelkezésünkre. Mindkettő lista zárt, fel kell iratkoznunk, mielőtt üzenetet küldhetnénk rájuk.

Az OSCAR 4.0 újdonságai

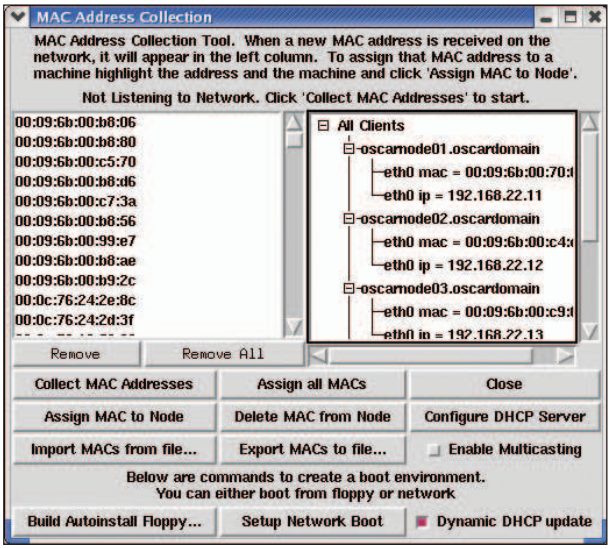
Sok újdonságot tervezünk beépíteni a hamarosan megjelenő új változatba, amelyek négy csoportra oszthatók: ***NEST***, csomópont-csoportok, ***Linux*** rendszercsomagok és SIS.

A ***NEST (Node Event and Synchronization Tools, csomópont esemény és szinkronizáló eszközök)*** azt biztosítja, hogy a fürt minden csomópontjának beállításai szinkronban legyenek a központilag tárolt adatokkal. Jelenleg új fürt-csomópont telepítésekor az ***OSCAR*** `post_install` parancsfájljait minden csomóponton le kell futtatni függetlenül attól, hogy erre ténylegesen szükség van-e. Bár ez a modell a közepes méretű fürtökön működőképes, nyilvánvalóan egy méretbeli korlátozást jelent. A legnagyobb eltérés a ***NEST***-ben, hogy a csomag-beállítások a kiszolgálóról töltődnek le, ahelyett, hogy az ügyfelekre töltődnének fel. A műveletek csak szükség esetén hajtódnak végre, ami sokkal elegánsabb megoldás a jelenleg alkalmazott feltétel nélküli végrehajtási sémánál.

A csomópont-csoportok a fürtben lévő csomópontok tetszőleges csoportosítását jelentik. Ezzel az új szolgáltatással lehetővé válik az ***OSCAR***-csomagok csoportonként eltérő kezelése. A következő változatban csak a kiszolgáló- és ügyfélcsomópont-csoportokat tervezzük támogatni, de a jövőben a felhasználóknak lehetőségük nyílik majd saját csoportok létrehozására is.



5. kép Itt adhatjuk meg a fürt csomópontjait és a hálózati jellemzőket



6. kép A fürt csomópontjain megtörtént a hálózatos rendszerindítás és a MAC-címek gépekhez rendelése

Az *OSCAR* meghatározó jellemzője a különböző Linux rendszercsomagok támogatása. Az új változattal reményeink szerint megvalósul a *Fedora Core 2* és *3*, a *Red Hat Enterprise Server 3.0* és a *Mandrake 10* támogatása. Ezzel együtt támogatni fogjuk az *IA-64* és *x86-64* rendszereket is. A *System Installation Suite (SIS, rendszertelepítő csomag)* amely magába foglalja a *SystemImager*-t is, az *OSCAR* képfájljait üzembe helyező programcsomag. Két továbbfejlesztés sorolható ide. Az egyik a lemeztípus önműködő észlelése. Hagyományosan az *OSCAR* képfájlok egyféle merevlemez-típusra készültek (vagy *IDE* vagy *SCSI*). Ezzel az *OSCAR*-kiterjesztéssel ugyanazt a képfájlt használhatjuk a különböző felépítésű és merevlemezzel rendelkező gépekhez feltéve, hogy a gép központi felépítése megegyezik.

Másodszor, hozzáférhető egy eszköz, amivel a felhasználók speciális rendszermag-modulokat használhatnak a csomópontok rendszerindításához. Időnként előfordul, hogy az újabb hardvereken nehezebb munkára bírni az *OSCAR*-t, mivel a *SIS* rendszermag-indító képfájl nem tartalmazza a támogatott meghajtóprogramokat. Ennek az eszköznek a segítségével használhatjuk a meglévő rendszermagot a biztosan működő modulokkal, és így az ügyfélcsomópontjainkra feltölthető a képfájl. Ez a szolgáltatás részét fogja képezni a *SystemImager* következő változatának és reményeink szerint az *OSCAR 4.0* verziónak is.

Csomagok létrehozása az OSCAR számára

Az *OSCAR* rendelkezik a megfelelő csomagokkal az általánosan használt, fürtözött környezetre felkészített programokhoz. Ezek egyszerű *RPM*-csomagok a megfelelő metafájlokkal és telepítő parancsfájlokkal, amelyeket az *OSCAR* elsődleges fejlesztőcsapata és a csomagírók készítenek és tartanak karban. Ha van egy olyan programunk, amelyet telepíteni szeretnénk a fürtre de az *OPD*-n (*OSCAR Package Downloader, OSCAR csomagletöltő*) keresztül nem találjuk, készítsünk hozzá egy csomagot. Az *OSCAR* fejlesztőcsapata nyitott a közreműködni szándékozók felé, és esetleg az általunk készített *OSCAR*-csomagnak is hajlandó helyet biztosítani. A csoport ezt a meghívást a programfejlesztőkre is fenntartja.

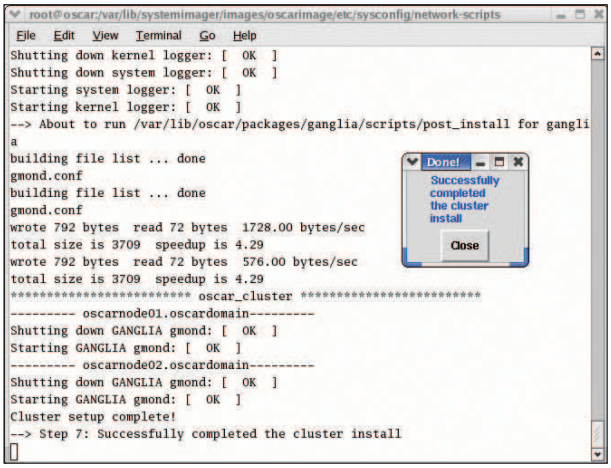
Az *OSCAR*-csomagok olyan decentralizált webhelyeken elhelyezkedő csomag-gyűjteményekben foglalnak helyet, amelyeket a csomagok szerzői biztosítanak a csomagok fájljainak tárolására. A tárhelyek címei egy központi tárhelylistán érhetőek el.

Az *OSCAR*-csomagok létrehozása általában egyszerű folyamat. Ha rendelkezésünkre áll egy kész *RPM*-csomag, a feladat felével már nem is kell foglalkoznunk, mindössze a csomag és *RPM*-fájlok metaadatait tartalmazó fájlokat kell létrehoznunk és néhány olyan parancsfájlt, amely gondoskodik a beállításokat tartalmazó fájlok terítésével a fürtön belül. A feladat végrehajtása viszonylag egyszerű, mert egy *OSCAR*-fürtön belül sok feltétel már eleve adottnak vehető. Amennyiben nem áll rendelkezésre megfelelő *RPM*-csomag, a folytatás előtt magunknak kell létrehoznunk az alkalmazás *RPM* formátumú alakját. Az *RPM*-csomag létrehozása a program összetettségétől függően lehet könnyű, de bonyolult is. Készítenünk kell egy leírófájlt és létre kell hoznunk az *RPM* fájl(oka)t és a hozzá tartozó *SRPM* forráscsomagot.

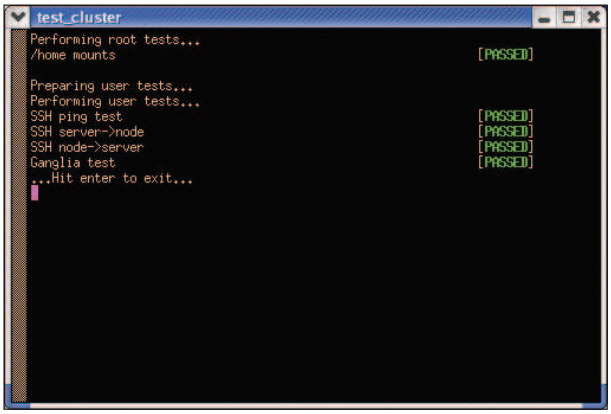
A *GSC (Genome Sciences Centre, Génkutató Központ)* által kiadott eső *OSCAR*-csomag a *Ganglia* csomagja volt, amely egy fürtmonitorozó rendszer. Már dolgozunk a második csomagunkon, amely a *Sun Grid Engine* nevű, a *Sun Microsystems* által támogatott nyílt forrású parancsfájl-ütemező rendszer. Ez a későbbiekben elérhető lesz majd az *OSCAR* letöltő rendszerén keresztül a *GSC OSCAR* tárhelyén.

Bioinformatikai alkalmazások és az OSCAR-fürt

A legelterjedtebben használt bioinformatikai program a genetikai kódsorok keresésére és illesztésére használt *BLAST*. A genetikai adatbázisban több genetikai kódsor keresése jól párhuzamosítható feladat, a folyamat nagyon könnyen



7. kép Beavatkozást nem igénylő lépés: végső telepítési beállítások és tisztító folyamatok végrehajtása



8. kép Minden rendszer működik, a fürt bevetésre kész

olyan alfeladatokra bontható, amelyek mindegyike a bemenetek egy adott csoportjával hajtja végre a keresést. Léteznek is szoftveres megoldások ennek a feladatszétválasztásnak a megvalósítására, ezek közül leginkább a nyílt forráskódú ***mpiBLAST*** és a kereskedelmi úton terjesztett ***Parcel*** érdemel említést. A ***BLAST*** párhuzamosított megoldásaival igen jó eredményeket lehet elérni, de természetesen elérhetünk egy olyan pontig, ahol már egyszerűen a csomópontok számának növelésével nem tudjuk tovább növelni a teljesítményt, mivel a kezdeti feladatszétválasztás munkatöbblete felemészti az így nyert teljesítményt.

Az ***FPC*** egy másik általunk sokat használt alkalmazás, amelyet ujjlenyomatok képének összeillesztésére, szerkesztésére és vizsgálatára használunk. Ennek első párhuzamosított változatát a *GSC* fejlesztette ki, ez azonban még nem támogatta a kötegelt feldolgozást. A közelmúltban építettük össze a párhuzamosított ***FPC***-t a ***Sun Grid Engine***-nel, így a felhasználók egyszerűen igényelhetnek bizonyos számú csomópontot a program futtatásához.

Dolgozunk egy bioinformatikai szolgáltatásokat megosztó egyenrangú alkalmazás, a ***Chinook*** kifejlesztésén is. Jelenleg ez még fejlesztési fázisban van, a saját fürtünkkel próbáljuk összeépíteni. Segítségével lehetőség nyílik majd a hálók összekapcsolására és ezzel a ***Globus*** eszközkészlet kiváltására.

Jelen pillanatban a 200 csomópontos fürtünk teljesítményét az emberi génállomány feltérképezése és osztályozása köti le. Az emberi génállomány hozzávetőlegesen 30.000 génből áll, az egyes géneket különféle algoritmusok segítségével vizsgáljuk. A géneket 1500-as csoportokba foglaljuk, amelyek egyszeri vizsgálata körülbelül négy napot venne igénybe egy 2 GHz-es ***Opteron*** gépen. Fürtök alkalmazása nélkül nem lenne lehetőség az ilyen kutatások folytatására.

Összegzés

Az ***OSCAR*** eszközkészlet hosszú utat tett meg az első kiadása óta. Egyre többen fedezik fel könnyű alkalmazhatóságát és egyszerű használatát s így egyre szélesebb körben élnek a fürtözés technológiájával. A bioinformatika mind nagyobb mértékben alkalmazza a nagyteljesítményű számításokat. Valószínű, hogy a bioinformatikus közösség számára egyre elérhetőbbé válik egy olyan megoldás, amely minden eszközt magában foglal majd a párhuzamos bioinformatikai alkalmazások egyszerű telepítéséhez és futtatásához.

Köszönetnyilvánítás

Köszönetet szeretnék mondani ***Mark Mayonak***, ***Asim Siddiquinek*** és ***Steven Jonesnak*** a lehetőségért, hogy a ***GSC Linux***-fürtjén dolgozhattam és felismerhettem az ***OSCAR***-ban rejlő lehetőségeket. Köszönet illeti továbbá az ***OSCAR*** fejlesztőcsapatát, a külső fejlesztőket és a felhasználókat, akik létrehozták ezt a nagyszerű közösséget a nagyteljesítményű számításokkal kapcsolatos tudás és információk megosztására. Végül, de nem utolsó sorban hálámat fejezem ki az ***NCSA*** munkatársainak, akik rengeteg időt és fáradságot fektettek az OSCAR projekt fejlesztésébe. Személyes köszönetemet fejezem ki ***Jeremy Enosnak***, ***René Warrennek*** és ***Martin Krzywinskinek*** a cikkhez fűzött értékes megjegyzéseikért és tanácsaikért.

Linux Journal 2004. november, 127. szám

Bernard Lee, a nagyteljesítményű számítások szakértője, a kanadai Michael Smith Génkutató Központ munkatársa, ahol feladata a Linux-fürtök kezelése és a bioinformatikai alkalmazások párhuzamos futtatási lehetőségeinek kidolgozása. Tagja az OSCAR központi fejlesztőcsapatának és rajongója a Sun Grid Engine-nek. A bli@bcgsc.ca címen érhető el.

KAPCSOLÓDÓ CÍMEK

- oscar.openclustergroup.org
- www.csm.ornl.gov/oscar04
- www.bcgsc.ca
- www.bcgsc.ca/gc/bomge/chinook

Bloglines webszolgáltatások (folytatás)

Néhány webes közösség gonosz módon bezárja a felhasználóit, a Bloglines ezzel szemben a nyitott megközelítés híve és megengedi, hogy saját parancsfájljainkkal irányítsuk a webszolgáltatás API felületét. Vágjunk bele és zárkózzunk fel kedvenc blogjainkkal.

Ezeket a sorokat néhány nappal a 2004-es november 2-án tartott amerikai elnökválasztás után írom. Mint közismert politika függőségben szenvedő, kiélvezhetem a számítógépesített modern érát, és az éppen aktuális bölcsességeket. Többé nem kell kapcsolgatnom a TV csatornák között, vagy újságokat olvasnom a helyi könyvtárban; nyugodtan végigkövethetem a részleteket amint a jelöltektől a sajtóhoz majd a különféle partizán oldalakra kerülnek. Lépést tartani a rengeteg különféle híroldallal igen sok időt vehet igénybe. Mint az elmúlt néhány hónapban láthattuk, mindenki nyert valamit a hírösszesítő programok használatával, amelyek begyűjtik a weblogok, újságok és más, gyakran frissített lapok által készített ***RSS*** és ***Atom*** tartalmakat. Az összesítő, mint a neve is mutatja, ezeket tartalmakat rendezi könnyen áttekinthető listába.

A ***Bloglines.com*** olyan Internetes kezdőpont amely web-alapú hírösszesítőt is kínál nekünk. Mindez önmagában még nem lepne meg senkit, hiszen a hírgyűjtemények, az összesítők és a web együtt szinte tálcán kínálják ezt a lehetőséget. Ráadásul a ***Bloglines*** nem is egyedi. Számtalan egyéb, igaz, talán nem annyira jól ismert Web-alapú hírösszesítő létezik. A ***Bloglines*** azonban egyedi szolgáltatást is kínál használóinak, ugyanis a ***Bloglines*** belső adatbázisát felhasználva saját összesítőt vagy akár saját alkalmazást készíthetnek a ***Bloglines*** által összegyűjtött adatokból. Mindez az információ ellenszolgáltatás nélkül hozzáférhető, egy viszonylag kötetlen szerződés mellett, valamennyi programozó számára, akit érdekel a ***Bloglines*** motor eredményeinek összegyűjtése. Mivel a ***Bloglines*** körülbelül óránként blogok és honlapok százezrein keres frissítéseket, a webszolgáltatás ***API*** használói biztosak lehetnek benne, hogy legfrissebb weblog tartalomhoz jutnak hozzá.

Legutoljára a ***Notifier API***-t vettük szemügyre, amely egy adott felhasználó elérhető, de még olvasatlan gyűjteményéhez enged hozzáférést. Megnéztük a ***Blogroll API***-t is, melynek segítségével a felhasználó, ha úgy kívánja, meghatározhatja vagy programjában felhasználhatja a gyűjteményre hivatkozó emberek listáját. Mit láthattuk, ezek a ***API***-k sokat segítettek, ha az új weblog bejegyzéseket szeretnénk megtalálni, vagy ha saját gyűjteményt listát akarunk összerakni az érdekes weblogokból.

Valami azonban hiányzott a cikkben bemutatott lehetőségek közül. Jó dolog tudni, hogy az új weblog bejegyzések a ***Bloglines*** kedvenceink közé kerül, de még jobb lenne ha azt is tudnánk, melyik blog frissült éppen. Továbbá, nem rossz, hogy lekérdezhetem az aktuális feliratkozási listámat, de még boldogabb lennék, ha azt is tudnám, melyikük frissült, és pontosan mikor frissültek legutoljára, valamint hány bejegyzés van az egyes weblogokban és azok a bejegyzések mit is tartalmaznak. Más szóval, a jelenlegi ***Bloglines*** felületet szeretném a sajátommal lecserélni, és az új weblog bejegyzéseket olyan alakban megjeleníteni, amely nem feltétlenül felel meg a ***Bloglines.com*** weblap által felajánlottnak. Szerencsére, a ***Bloglines*** webszolgáltatás fejlesztői lehetővé tették, hogy pontosan így járjunk el, mégpedig a ***sync API*** segítségével. Ebben a hónapban, folytatjuk a ***Bloglines*** webszolgáltatásainak felfedezését, és elmerülünk a ***sync API*** részleteiben. Megpróbálunk létrehozni egy saját, egyszerű hírösszesítőt, amely rendelkezik néhány, a ***Bloglines*** felületből már ismert képességgel.

Feliratkozások és elemek

Végső soron a ***Bloglines*** és a hozzá hasonló hírösszesítők nem állnak másból, mint egy ***URL*** listából. A két hónappal korábban összerakott, ***Python*** nyelvű és ***Universal Feed Parser*** alapú hírösszesítőnk pontosan ilyen program volt – ***URL*** listát vizsgált egy fájlban és az adott ***URL***-ekhez tartozó legfrissebb tartalmat gyűjtötte össze. Minden egyes weblog küldemény a lista egy-egy ***URL***-jének felel meg. Ha az ***URL***-t eltávolítjuk a feliratkozási listánkból, a hozzá tartozó küldemények többé már nyilvánvalóan nem érdekesek a felhasználó számára, ezért nem is látja őket.

Az a tény, hogy a ***Bloglines*** több felhasználóval dolgozik, azt jelenti, hogy nem elég egyetlen ***URL*** listát tárolnia, hanem azt is meg kell jegyeznie, hogy melyik ***URL*** melyik felhasználóhoz tartozik. Bár ez nyilván kicsit megbonyolítja a dolgokat, a modern magas szintű nyelvek segítségével nagyon könnyű megérteni a két adatszerkezet közötti különbségeket. Az ***URL*** lista egyszerű tárolása helyett egy hash táblát kell létrehoznunk, ahol a felhasználói azonosítót használjuk kulcsként értéke pedig az adott felhasználóhoz

1. lista A felhasználó feliratkozásainak megjelenítése

```
#!/usr/bin/perl
use strict;
use diagnostics;
use warnings;
use WebService::Bloglines;
my $username = 'reuven@lerner.co.il';
my $password = 'MYPASS';
my $bloglines =
  WebService::Bloglines->new(username =>
$username,
                             password =>
$password);
# Olvasottként akarjuk megjelölni?
my $mark_unread = 0;
# Milyen dátumtól kezdve szeretnénk letölteni?
# (ennek Unix időnek kell lennie)
my $subscriptions = $bloglines->listsubs();
if ($subscriptions)
{
    # a tartalmak listája
    my @feeds = $subscriptions->feeds();
    # Beszerezzük a tartalmak címét és URL-ét
    foreach my $feed (@feeds) {
        my $title  = $feed->{title};
        my $url    = $feed->{htmlUrl};
        my $subId  = $feed->{BloglinesSubId};
        print "Subscribed to '$title', "
            . "subId '$subId' at '$url'\n";
    }
}
else
{
    print "No subscriptions.\n"
}
```

tartozó lista lesz. Ha a felhasználó egyedi azonosítóját ismerjük, könnyedén nyomon tudjuk követni az adott felhasználó feliratkozásait.

Természetesen, a ***Bloglines*** nem néhány ezer ember listáit tartja karban, hanem sok tíz vagy százezerét. Így nyugodtan állíthatjuk, hogy nem ilyen naiv megvalósítást használnak, amely legfeljebb egy kísérlet, vagy néhány ember számára tervezett összesítő számára felelhet meg. A dolgok kicsit trükkösebbé válnak amikor megközelítjük a ***Bloglines*** felhasználói terhelését. A felhasználók feliratkozási listái nem lehetnek egyszerű ***URL***-ek; sokkal valószínűbb, hogy az ***URL***-ekhez rendelt azonosító számokat (vagy ahogy az adatbázis körökben nevezik „elsődleges kulcsot") használnak. Egy ilyen rendszernek megvan az az előnye, hogy több jelentkezőnek is lehetővé teszi a feliratkozást a lap hírösszesítőjére, a ***Bloglines*** a már meglévő feliratkozások alapján javaslatot tud tenni nekik , hogy mely további weblogok érdemesek még a figyelmükre.

Ezek után nem okoz nagy meglepetést ha megtudjuk, hogy az új weblog küldemények lekérése a ***Bloglinestól*** két lépésből áll. Az első lépésben lekérjük a feliratkozások listáját, pontosabban a ***Bloglinestól*** lekérdezzük a felhasználóhoz rendelt feliratkozási azonosítókat. Ezt követően utasítjuk a ***Bloglinest***, hogy küldje el az adott felhasználóhoz és a kapott azonosítóhoz tartozó összes új elemet.

A ***Bloglines*** webkiszolgáló ***API*** megvalósítása több különféle programnyelven is elérhető. Minthogy az új alkalmazásokat többnyire ***Perlben*** készítem, a `WebService::Bloglines` modult fogom használni amelyet feltöltöttek a ***CPAN***-ra, azaz a ***Comprehensive Perl Archive Network*** nevezetű világméretű web és ftp kiszolgáló hálózatra, amelyről a ***Perl*** és annak moduljai tölthetőek le. Az *1. listában* látható egyszerű program (***bloglines-listsubs.pl***) megjeleníti a címet, a feliratkozás azonosítóját és a hozzá tartozó ***URL*** a felhasználó összes feliratkozására. Minden feliratkozáshoz további információ is elérhető; ezeket a `WebService::Bloglines`, illetve a ***Bloglines API*** dokumentáció részletesen ismerteti.

Amennyiben meg szeretnénk őrizni a felhasználók ***Boglines.com*** felületén látható sorrendjét , érdemes a `folders` függvényt használnunk az 1. listában olvasható `feed` függvény helyett. A `feed` ugyanis egyszerűen csak visszaadja a feliratkozásokat, a `folders` ellenben ugyanolyan szerkezetben adja őket vissza, ahogy azt a Bloglines lapján láthattuk.

Feliratkozások letöltése

Most már tudjuk, hogyan szerezhetjük meg egy adott ***Bloglines*** felhasználóhoz tartozó feliratkozás azonosítókat, így le tudjuk kérdezni a feliratkozás azonosítókhoz tartozó egyes elemeket is. Ezt mutatja be a *2. lista* rövid programja amely letölti a felhasználó összes feliratkozását, majd valamennyihez megmutatja a legutóbb frissült elemeket. A kimenete nem ***HTML***, hanem egyszerű szöveges formátum, következésképpen a megjelenített hivatkozásokra nem tudunk rákattintani. Mindazonáltal nem különösebben nehéz egy ilyen programot ***cron*** feladatként lefuttatni és a kimenetét egy ***HTML*** fájlba irányítani, így percre friss, személyre-szabott tartalomlistát kapunk. Természetesen a ***Bloglines*** ingyenesen biztosítja nekünk ezt a szolgáltatást valahányszor csak belépünk a weblapjukra. Így aztán a program érdekes tesztje a ***Bloglines*** webszolgáltatásainak, de azokhoz képest igazából nem nyújt semmi újdonságot.

Az egyik ügyes húzás amit a ***Bloglines*** a webszolgáltatás definíciójában bevezetett , hogy ***HTTP*** visszatérési kódokat használ a hibák és szokatlan események visszajelzésére. Például, a `200 (OK)` válaszkód azt jelzi, hogy van új beolvasandó elem illetve, hogy a `getitems($subId)` egy vagy több ilyen elemet tartalmaz. A 304 (változatlan) válaszkód amely egyébként azt jelzi, hogy a ***HTML*** az utolsó lekérdezés óta nem változott, itt egy kicsit más szerepet kap; azt jelenti, hogy a az adott feliratkozó már az összes elemet látta ebben a feliratkozásában. A többi válaszkód (401, 403 és 410) azonosítási hibákat jelez, ami valószínűleg annak következménye, hogy a kérelmező felhasználó elgépelte a ***Bloglines*** felhasználónevét, jelszavát, esetleg mind a kettőt.

Sajnos ***Perl*** alatt e hibakódok kezelése messze van az optimálistól. Lekérdezésükhöz az `eval` blokkon belül meg kell hívnunk a `$bloglines->getitems()` függvényt, majd közvetlenül az `eval` meghívását követően meg kell vizsgálnunk a `$@` értékét. Ha a `$@` üres, feltételezhetjük, hogy 200-as (OK) HTTP válaszkódot kaptunk és vannak beolvasandó új elemek. Amennyiben azonban értéket tartalmaz, újra kell írnunk a kimeneti üzenetet, ahogy azt a 2. listában meg is

2. lista bloglines-getitems.pl

```
#!/usr/bin/perl
use strict;
use diagnostics;
use warnings;
use WebService::Bloglines;
my $username = 'reuven@lerner.co.il';
my $password = 'MYPASS';
my $bloglines =
  WebService::Bloglines->new(username => $username,
                             password =>
$password);
# Olvasottként akarjuk megjelölni?
my $mark_unread = 0;
# Milyen dátumtól kezdve szeretnénk letölteni?
# (ennek Unix időnek kell lennie)
my $subscriptions = $bloglines->listsubs();
if ($subscriptions)
{
    # a tartalmak listája
    my @feeds = $subscriptions->feeds();
    foreach my $feed (@feeds) {
        my $title  = $feed->{title};
        my $url    = $feed->{htmlUrl};
        my $subId  = $feed->{BloglinesSubId};
        print "Subscribed to '$title', "
            . "subId '$subId' at '$url'\n";
    my $update;
    # Elkapjuk a hibákat !
    eval {$update = $bloglines->getitems($subId);};
    # Figyeljük a hibákat, kiírjuk ha nincs változás.
    if ($@) {
        if ($@ =~ /^304 No Change/) {
            print "\t No change\n";
        }
        else {
            print "\t Error code '$@' "
                . "retrieving updates.\n";
        }
    }
    # Nincs hiba?  Néhány alap dolgot kiírunk az
elemekről.
    else
    {
        foreach my $item ($update->items)
        {
        my $title       = $item->{title};
        my $creator     = $item->{dc}->{creator};
        my $link        = $item->{link};
        my $pubDate     = $item->{pubDate};
        print "\t$title by $creator "
            . "on $pubDate ($link)\n";
        }
    }
    }
}
else
{
    print "No subscriptions.\n"
}
```

tettük. Amennyiben ezt a metódushívást nem tesszük `eval` blokkba, akkor sajnos a programunk végzetes hibával leáll, amint a 200-as válaszkódon kívül bármi mást kapunk.

Végül a ***Bloglines*** funkcióit két opcionális paraméter teszi teljessé. Az első n-ként ismert paraméter egyszerű igaz-vagy-hamis érték (1 vagy 0) amely értesíti a ***Bloglines***-t , hogy frissítenie kell-e a „már láttam" bitet az éppen letöltött cikkeknél. Egyébként amikor a felhasználó a ***Bloglines.com*** webes felületén keresztül nézi a weblog küldeményeit ezt mindig 1-re állítjuk, ami azt jelenti, hogy a korábban elolvasott cikkek újra már nem kerülnek elő. Talán, mivel tudták, hogy a webszolgáltatások ***API*** más hírösszesítő alkalmazásokat is kiszolgál, a Bloglines bölcsen 0-ra állította ezen érték alapértelmezését az ***API***-ban.

A második, d nevű opcionális paraméter azt az első dátumot mutatja meg a ***Bloglinesnak*** amelytől kezdve az adott honlap cikkeit le szeretnénk tölteni. Az érték ***UNIX*** időformátumban értendő, azaz 1970 január elseje óta eltelt másodpercek számát kell megadnunk. Ezt a számot a nagyobb nyelvek mindegyikében készen kapjuk, és segítségével nagy pontossággal meg tudjuk adni, hogy pontosan milyen mélyen szeretnénk az adott honlap ***Bloglines*** tárolta történelmébe nyúlni.

Összefoglalás

Az igazat megvallva lelkes ***Bloglines*** felhasználónak tartom magam, annak ellenére, hogy pontosan tudnám hol is van a lap és a cég székhelye. Nehezen tudom elképzelni, hogy továbbra is ingyenes és reklámmentes marad, hacsak fejlesztői nem jótékonyságból cselekszenek vagy végzetesen naivak. Kedvelem a kiváló felületét, a tényt, hogy könnyedén el tudom érni azokat a weblogokat amelyektől politikai éleslátásom függ (vagy éppen mulatságom, attól függően ki hogyan ítéli meg az ilyen bölcsességeket) és gyors, hibatűrő képességeit.

Azonban ahogy az ***Amazon***, ***eBay*** és ***Google*** megmutatta az elmúlt néhány évben, nyers adatainkhoz webszolgáltatás felületet adva olyan új kreatív alkalmazásoknak nyitunk kaput, amelyekre a belső fejlesztők álmukban se gondoltak volna. A ***Bloglines*** mostanában kezdte el funkcióit webszolgáltatások formájában közzétenni, és igaz ugyan, hogy mindez egyelőre csak kezdeti és kísérleti lépés, a magam részéről nagyon ígéretesnek látom. Alig várom, hogy olyan alkalmazásokat lássak, amelyek erre az ***API***-ra valamint a ***Bloglines*** és versenytársai által kiadott további API felületekre épülnek, hogy a ***Bloglines*** lapját tegyék a weblogok, olvasók és fejlesztők Mekkájává.

Linux Journal 2005. március, 131. szám

Reuven M. Lerner veterán web/adatbázis tanácsadó és fejlesztő, jelenleg végzős hallgató a Northwestern University Learning Sciences programjában. Weblogja a altneuland.lerner.co.il címen olvasható őt magát pedig reuven@lerner.co.il címen érhetjük el.

Az Oddmuse wiki motor

A wikik mókásak és nagyon hasznosak. Próbaképpen telepítsük Alex Schroeder Oddmuse programját.

Az első wiki weboldalt, a ***WikiWikiWeb***-et, 1995-ben ***Ward Cunningham*** hozta létre a ***Portland Pattern Repository Project*** számára. Az azóta eltelt idő alatt számos wiki motort fejlesztettek különböző programnyelveken. Az egyik ilyen ***wiki motor Alex Schroeder Perl*** nyelven íródott ***Oddmuse*** programja.

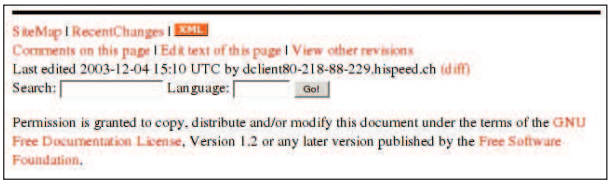
Mi a Wiki?

Bár wikik közel egy évtizede léteznek, úgy tűnik népszerűségük töretlenül növekszik, és sok ember még csak most találkozik a wiki-koncepcióval. Ezért talán érdemes egy rövid áttekintést adnunk a wikikről. Aki már jól ismeri a wikiket, nyugodtan a következő fejezetre ugorhat.

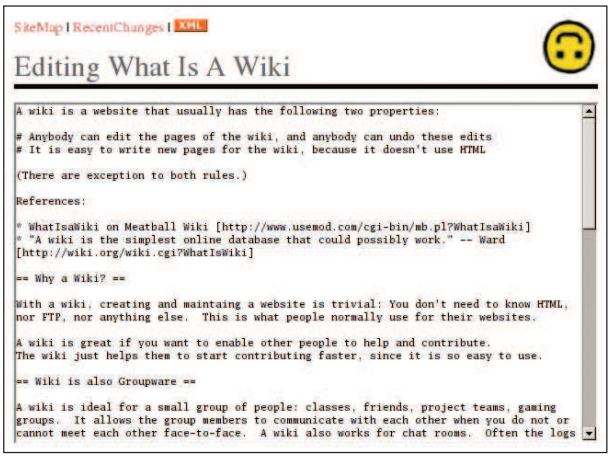
A wiki alapelve egyszerű: olyan weboldalról van szó, amely minden felhasználó számára lehetővé teszi, hogy kizárólag a böngészőt használva bővítse vagy szerkessze azt. Ez az egyszerű elképzelés nagyon hatásosnak bizonyult, hiszen rengeteg önkéntes fejlesztőnek teszi lehetővé, hogy közös munkával készülő weboldalakat hozzanak létre. A wiki azonban kisebb csoportok vagy magányos szerzők számára is nagy segítség, akik egyszerűen szervezhetik és hozhatják létre információikat. Ahogy egyszer ***Cunningham*** mondta: „[A wiki] a legegyszerűbb on-line adatbázis ami egyáltalán működhet.” Tapasztalataim szerint a wikik nagyon hasznos eszköznek bizonyulhatnak. Segítségükkel projekteket szervezhetünk, és dokumentációt készíthetünk szinte bármiről, meglepő módon sokkal fájdalommentesebben és kevésbé időrabló módon mint egyébként. Talán a legegyszerűbben úgy érthetjük meg a wiki lényegét és működését, ha megnézzük milyen minőségű dokumentumok hozhatók létre, és felkeresünk egy felnőtt, bejáratott wiki lapot. Kiváló példa erre a ***Wikipedia***.

A ***Wikipedia*** egy enciklopédia, éppen olyan mint egy többkötetes, kinyomtatott-bekötött enciklopédia. A bekötött enciklopédiáktól eltérően azonban a ***Wikipedia*** hatalmas „szerkesztői gárdával” rendelkezik amelynek bármely böngészővel rendelkező Internet felhasználó tagja lehet. Ezen kívül minden egyes cikk szövege a megfelelő helyeken hivatkozásokat tartalmaz más ***Wikipedia*** cikkekre. A lap látogatói cikkeket hozhatnak létre , átszerkeszthetik a meglévőket. Változtatásaik és bővítéseik pedig azonnal hozzáférhetőek lesznek a többi felhasználó számára. A változásvezető rendszer segítségével ezek a módosítások könnyedén elmenthetők, ami oltalmat nyújt a tévedésből elkövetett vagy rossz szándékú módosításokkal szemben. Hogy fogalmat alkothassunk, mennyire könnyen vihetünk tartalmat a wikibe, képzeljük el, hogy valamilyen wiki weboldalon éppen a rágcsálókról olvasunk cikket. Feltűnik nekünk, hogy bár igazán szépen megírt cikk, elfelejtett említést tenni a világ legnagyobb ma is élő rágcsálójáról, a vízidisznóról. A figyelmetlenséget javítandó, rábökünk a lap hivatkozásai közt található Edit szövegre (miután kitöltöttük a szerkesztéshez szükséges szabványos Web-űrlapot). Ezáltal be tudjuk gépelni a vízidisznóról szóló mondatunkat, és amint kimentettük a módosításokat, a cikk új verziója máris mindenki rendelkezésére áll. Ilyen egyszerű.

Tegyük fel, hogy a cikkben elhelyeztünk egy hivatkozást ***Dél-Amerikára***, hiszen ez a vízidisznó élőhelye és azt szeretnénk, ha a ***Dél-Amerika*** szó a ***Dél-Amerikáról*** szóló cikkre

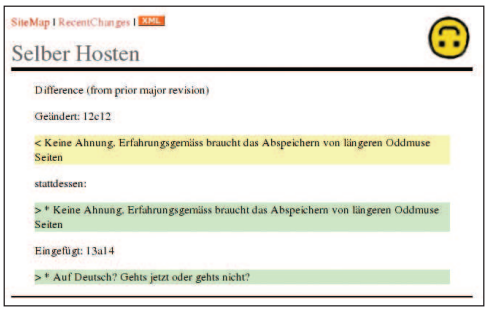


1. ábra Jellemző Oddmuse lapléc, a megszokott wiki funkciókkal: a lap szövegének szerkesztése és keresés

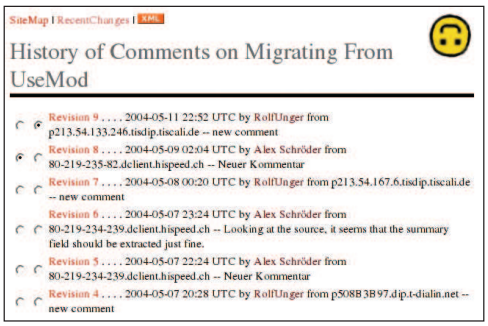


2. ábra A lapformátum jellemző szerkesztőszövege. A képernyőképen nem látszanak a Preview (előnézet) és Save (mentés) gombok.

mutatna. A hivatkozás létrehozásához mindössze az egyszerű wiki csatolási szabályt kell ismernünk (ha a lap *Oddmuse*-t használ, a *Dél-Amerika* szót dupla szögletes zárójelbe kell helyeznünk) és a lap elkészítésekor a wiki motor automatikusan létrehozza a *Dél-Amerikáról* szóló cikkre mutató hivatkozást. Amennyiben nincs még *Dél-Amerikáról* szóló cikk, a wiki motor létrehoz egy űrlapot, ahol bárki elkészítheti a hiányzó cikket. Így aztán a wiki cikkről-cikkre bővül és egyre hasznosabb lesz.
Egy másik példa, amely jól mutatja milyen hasznos tud lenni a wiki kis csoportok számára is, az általam és üzleti partnereim által használt oldal. Ezt műszaki információk rögzítésére, például kiszolgáló beállítások megváltoztatására, projekt dokumentálásra, programtelepítésre, általános feladatok szokásos megoldására és egyéb feladatokra használjuk. Ezen követjük nyomon üzleti adatainkat is, például az ügyfelek adatait.
Az előnyök teljesen nyilvánvalóak számunkra. Az elmúlt idők tapasztalatai alapján meg kellett állapítanunk, hogy a rendszeradminisztráció és projektdokumentáció lényegesen nagyobb része került rögzítésre amióta wiki-t használunk. Ezen felül az adatok kereshetőek és könnyen szerkeszthetőek. Ráadásul, mivel a cikkváltozásokhoz és megjelenésekhez *RSS* tartalmat is szolgáltatunk, minden érintett könnyedén nyomon követheti az eseményeket.



3. ábra Az Oddmuse jól olvasható eltérés-megejelnítője kiválóan mutatja mi változott meg az egyes verziók között



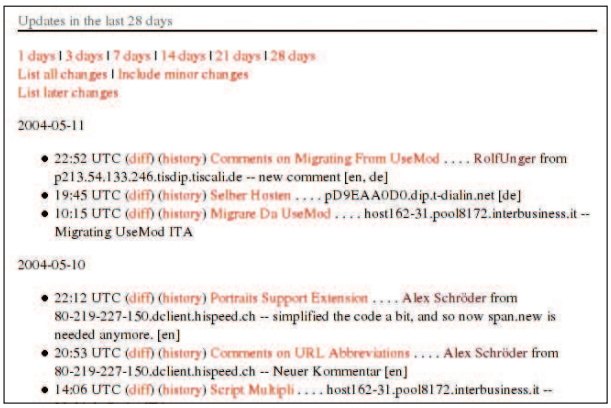
4. ábra Jellemző történet lap. A jelölőnégyzetek segítségével a felhasználó beállíthatja mely lapok közötti különbségekre kíváncsi.

Oddmuse képességei

Az *Oddmuse* egyetlen *Perl* script. Ezzel a telepítés jelentősen leegyszerűsödik és egyben megfelel a *Schroeder*-projekt egyik céljának: az *Oddmuse* legyen egyszerű és könnyen használható. A rugalmasság elvesztését az *Oddmuse* bővíthetősége ellensúlyozza. Bizonyítékként, a projekt honlapján megtalálhatjuk az elérhető modulok listáját.
A szokásos, kötelező wiki elemek mellett (utolsó változtatások automatikus követése, hatékony hivatkozási jelölésrendszer, verziókezelés és az egyes változatok különbségeinek vizuális megjelenítése,) az *Oddmuse* számos további figyelemre méltó képességgel és előnyös tulajdonsággal rendelkezik. Az egyik ilyen dolog ami mély benyomást tett rám, az *Oddmuse* dokumentáció minősége. *Schroeder* nagy figyelmet fordít a különféle nyelveken megjelenő bőséges dokumentációra és ennek meg is van az eredménye. Stílusosan az éppen dokumentált eszközt használja a dokumentáció karbantartására és készítésére. Az *Oddmuse* projekt weblapjára látogatva mindjárt a motort is megtekinthetjük működés közben, és bőséges információt találunk arról, hogyan állíthatjuk be és szabhatjuk testre.
Olyan ember lévén, aki a szeret megfelelni a *World Wide Web Consortium (W3C)* szabványainak, örömmel láttam, hogy az *Oddmuse* érvényes *HTML 4.01 Transitional* lapokat készít. Kiegészítésként érdemes megjegyezni, hogy szépen együttműködik a *Cascading Style Sheets (CSS)* technikával is. Ha szeretnénk hírgyűjteményt kiadni *RSS*-en keresztül, az *Oddmuse* egyaránt támogatja a wiki lapunkhoz készített kimenő *RSS* tartalmat és a hírgyűjtemény készítésére szolgáló bejövő *RSS* tartalmakat.
A wiki lapok egyik nagy kérdése a vandalizmus. Végül is mi gátolhat meg valakit abban, hogy egy honlapot szerkesztve valamilyen módszerrel teleszemetelje azt? Amennyiben meg szeretnénk őrizni a wiki egyik legnagyobb előnyét, nevezetesen, hogy bárki képes lapokat szerkeszteni és létrehozni, kénytelenek vagyunk nyitottan hagyni a lapokat és magunk helyreállítani az elrontott oldalakat. Szerkesztők figyelmes csoportja a *RecentChanges (friss változások)* lapon könnyen felderítheti a vandalizmus nyomait. Amennyiben viszont hajlandóak vagyunk lemondani a nyílt szerkesztés előnyeiről és inkább a vandalizmust szeretnénk megfékezni, az *Oddmuse* két módszert is kínál erre. Lezárhatunk adott oldalakat vagy akár az egész honlapot, illetve jelszavakkal biztosíthatjuk bizonyos személyeknek a szerkesztési hozzáférést. Másik lehetőség, hogy egyszerűen kitiltjuk a problémás felhasználókat.
Az *Oddmuse MySQL* vagy *PostgreSQL* adatbázis helyett egyszerű állományokat használ adattárolásra. Sok érvet lehet felhozni amellett, hogy miért előnyös ez, és épp ilyen sok érv szól amellett is, hogy miért hátrányos. Egy részről sima állományokat használni adatbázis helyett adminisztrációs szempontból mindenképpen egyszerűbb. Más részről az adatbázisoknak is megvannak az előnyei, főként az adattárolás és visszakeresés terén. Ha úgy érezzük wiki oldalunkat kizárólag adatbázis háttérrel tudjuk elképzelni az *Oddmuse*-t nem nekünk találták ki. De ha csak a sebesség miatt aggódunk, érdemes megemlíteni, hogy a keresés meglehetősen gyors. Jó példa erre az *Oddmuse* alapú *Emacs Wiki*, amely jelenleg 2,242 lapot tartalmaz.

Telepítés

Az *Oddmuse* weblap telepítésről szóló részében a telepítéssel és beállításokkal kapcsolatos legfrissebb parancsfájlokra és útmutatókra találunk hivatkozásokat. Az alapbeállítás rendkívül egyszerű. Miután letöltöttük a parancsfájlt, egyszerűen helyezzük el a *CGI* könyvtárunkba és tegyük végrehajthatóvá. Azon nyomban működnie kell. Ha nem így történik nézzünk utána a dolognak az *Oddmuse* hibaelhárító oldalain. Még ha nem is tervezzük a wiki átalakítását, egy dolgot mindenképp érdemes elvégezni mielőtt bármi máshoz kezdenénk. Nevezetesen a parancsfájlban módosítanunk kell



4. ábra A friss változásokat követő lap jellemző kinézete

a ***$DataDir*** változót, hogy annak a könyvtárnak az abszolút útvonalát tartalmazza ahol az ***Oddmuse*** majd az adatait tárolni fogja. Ha ezt a változót nem írjuk át, az ***Oddmuse*** a ***/tmp/oddmuse*** könyvtárat fogja adatkönyvtárként használni, így könnyen elveszthetjük adatainkat.
A másik dolog amit érdemes megtenni, függetlenül attól, hogy milyen egyszerű vagy bonyolult wiki beállítást szeretnénk összehozni, az adminisztrátor és a szerkesztő jelszavának beállítása. A megváltoztatandó változók a ***$AdminPass*** és ***$EditPass*** lesznek. Mivel a jelszavak egyszerű szöveges állapotban tárolódnak, a wiki lapunk biztonsága éppen annyira erős, mint a kiszolgálóé általában. Ettől függetlenül érdemes erős jelszavakat választani.
Bár az alaptelepítés teljesen jól működő ***Oddmuse*** wiki lapot készít, melyen valószínűleg mindent megtalálunk amire szükségünk van, esetleg érdemes fontolóra venni egy komolyabb telepítést is. Magam részéről a komolyabb telepítést választottam a mostanában feltett ***Oddmuse*** wiki lapjaimhoz, és úgy vélem, a kapott rugalmasság bőségesen megéri a beállításokkal járó plusz fáradtságot.
Először is a weblapon egyértelműen leírt csomagoló-parancsfájl módszert használtam. Ezzel a módszerrel nem szükséges minden parancsfájl-frissítés után újra megadni az adatkönyvtár nevét. Másodszor külső beállításfájlt használtam, azon belül pedig külső ***CSS*** állományt adtam meg.

Beállítások

Az ***Oddmuse*** wiki lapunk most már működőképes, úgyhogy tetszés szerint állítgathatjuk. Néhány beállítást magában a wikiben is elvégezhetünk,például zárolhatjuk a lapokat vagy, ha nem szeretnénk nyílt wikit üzemeltetni, akár az egész honlapot.
A beállítások megváltoztatásához és egyéb wiki adminisztrációs feladatok elvégzéshez adminisztrátorként kell azonosítani magunkat. Ehhez viszont meg kell látogatnunk a jelszólapot. A jelszólapra nincs külön hivatkozás, ezért nekünk kell begépelni a következő ***URL***-t:
http://www.example.com/wiki/current.cgi?action=password

Természetesen saját gépünk nevét és saját elérési utunkat kell használni. A fenti ***URL*** példában az ***Oddmuse CGI*** programot azért nevezik ***current.cgi***-nek mert a csomagoló parancsfájl telepítési útmutatót követtük. Miután azonosítottuk magunkat, tetszés szerinti beállításokat végezhetünk a lapok láblécében megjelenő különleges menüpontokat használva. További változtatások közvetlenül a parancsfájl módosításával végezhetők el. Számos programváltozót igazíthatunk az igényeinkhez. Ha külső beállítás-állományt szeretnénk használni, a programváltozókat természetesen ott tudjuk átállítani. A weblapon részletes leírást találunk minden programváltozóhoz. Ezekkel a változókkal vezérelhetjük például a wiki nevét, a használt stíluslapot, a lapok láblécében megjelenő jelvényünk ***URL***-jét és egyéb értékeket.
Nagyon örülök, hogy ***Schroeder*** beépítette a ***CSS*** támogatást az ***Oddmuse***-ba, hiszen ezáltal sokkal könnyebben tudjuk megváltoztatni a wiki kiosztását és kinézetét mint egyébként. Miután a wiki elkészült és futásképes, néhány percet szántam a stíluslap alakítgatására, míg végül el nem értem a nekem tetsző wiki kinézetet.
Ezen kívül a modulok és kiterjesztések beépítésével további jelentős változtatásokat vezethetünk be a wikinkbe, sőt akár saját változatot is készíthetünk. Az ***Oddmuse Weblapon*** megtaláljuk a modulok és kiterjesztések teljes listáját. Ha úgy tetszik az ***Oddmuse***-t akár blogrendszerré is formálhatjuk.

A Wiki életrekeltése

Amikor elégedettek vagyunk a wiki telepítésével és beállításával, elkezdhetjük a lapokat szerkeszteni. Mielőtt nekifognánk azonban, mindenképpen javaslom, hogy olvassuk el a az ***Oddmuse*** weblap ***szövegformázási szabályok (Text Formatting Rules)*** fejezetét. Miután megismertük a szabályokat, a lapok létrehozása és szerkesztése nem nehéz feladat, de felbecsülhetetlenül hasznos ha tudjuk mire képesek ezek a szabályok.
Egy új wiki alapítóiként, többre lesz szükségünk egyszerű műszaki részletek ismereténél. A wiki sikere az alkotók jelentkezésétől függ, így a közösségi szempontokat legalább annyira figyelembe kell venni mint a műszaki szempontokat. Ez jelentősen megbonyolítja a dolgokat. Végül is a közösségeket nem tölthetünk le tarlabdaként make-fájllal! Nekünk kell jelentkezésre ösztönöznünk és lelkesítenünk az embereket majd nevelgetnünk a wiki körül növekvő közösségünket. Ez időrabló foglalatosság és aligha adható rá pontos recept. Szerencsére a wikik alapítói vették maguknak a fáradtságot és megírták tapasztalataikat. Kiváló forrás a ***MeatballWiki*** honlap ***WikiLifeCycle*** oldala (melyet a források között megtalálunk). Ebből megtudhatjuk milyen módszerekkel gyűjthetünk szerzőket, hogyan válasszunk nevet, jelöljük ki a határokat, miképpen válasszuk meg a wiki célját vagy küldetését, hogyan alakítsuk ki a viselkedési elvárásokat, előzzük meg a pangást és így tovább. Most, hogy felvértezték magunkat egy jól beállított wiki oldallal és megértettük a közösségi kérdések alapjait, már semmi sem állíthat meg bennünket egy virágzó wiki oldal felé vezető úton.

Linux Journal 2005. március, 131. szám

Brian Tanaka 1994 óta foglalkozik UNIX rendszeradminisztrációval és olyan cégeknél dolgozott mint a The Well, SGI, Intuit és a RealNetworks. A btanaka@well.com címen érhető el.

Linux-sokszorosítás iskolai gépekre

Középiskolánkban sikerült egy UHU-linuxra mindazon alkalmazásokat feltelepítenem, amelyekről úgy gondoltam, hogy hasznosak lehetnek a diákok számára, majd az így kialakított rendszer sokszorosítására kidolgoztam egy egyszerű módszert.

A szokásos *OpenOffice.org*-on, gépíróprogramokon (*tpgt*, *ktouch*), és *adblock*-kal illetve *webdeveloperrel* kiegészített *Firefox* böngészőn túl – leginkább a különböző szakkörökre járók kedvéért – telepítettem még néhány hasznos alkalmazást. Ilyen a *Xored* cég által módosított (szabad programként letölthető) *Eclipse* fejlesztői környezet, valamint a *NuSphere PhpEd* nevű programja, amely PHP programok nyomkövetését teszi lehetővé. (Az *Eclipse* megvan *.uhu* csomagban is, de nem sikerült úgy beállítanom, hogy lehessen PHP-t is futtatni benne.) *Kdevelop* is felkerült, de ebben a PHP-hoz nem találtam nyomkövetőt. Természetesen a *Kylix* sem maradhatott el.
Iskolánk egy pályázat keretében néhány éve megvette a *Maple7* programot. Ez, az akkori technikai szintnek megfelelően *Windows 98* alatt fut, *Windows XP* alatt azonban már nem. A viszonteladó cég széttárta karjait: sajnos nincs mit tenni, meg kell venni az új változatot. Szerencsére a telepítő CD-n található linuxos változat még mindig nagyszerűen működik. A *Crossover Office* is gond nélkül indítja például az oktatáshoz szükséges *Comenius Logo*-t. Az iskolánkban kiosztott *TI92*-es grafikus kalkulátorokat össze lehet kötni számítógéppel a *TI-link* program segítségével. Webkiszolgálót, valamint *postgresql*, *mysql* és *firebird* adatbáziskezelőket is feltettem, bár ezek nem indulnak automatikusan. A *DBDesigner4*-gyel grafikus felületen lehet adatbázist tervezni, sőt, adatbázishoz kapcsolódva ki lehet rajzoltatni annak szerkezetét. A programnak ugyan nincs *Postgresql* kiviteli lehetősége, de a *www.osb.hu/z/My2Pg* Perl szkript elvégzi ezt a konverziót.

Előkészületek a klónozáshoz

Nem érdemes 5 GB-nál kisebb helyet adni az *UHU Linuxnak* (ez lehet logikai partíció is). Én 3.6 GB-tal kezdtem, de kicsinek bizonyult. Jó esetben van egy bőséges helyet biztosító *Samba* kiszolgáló is a hálózatban.
A képmásfájl készítése során igen jó szolgálatot tett a *SystemRescueCD*, amely a *Linuxvilág* mellékleteként nemrég megjelent, de le is tölthetjük a webhelyéről. Ezen megtalálható a képmás-fájl elkészítéséhez és visszaállításához remekül használható `partimage` program. (Erre a célra más jó programok is léteznek. Ilyen például a `dump`/`restore` páros.). A képmás-fájl mentéséhez a `partimage` programot interaktív üzemmódban használtam, ellentétben a visszaállítással, amit parancssorból, automatikusan érdemes végrehajtatni. Az interaktív használat annyiban jó, hogy az esetleges mulasztásokat (például kiírandó partíció lecsatolása) kedvesen közli velünk, és lehetőségünk van magyarázó szöveget is írni az elmentett partícióról, ami jó lehet olyankor, ha már sok próbálkozásunk volt, és valamelyik korábbi (ismert) fázisba szeretnénk visszatérni.
Sajnos a *bz2* tömörítményként való mentés valami miatt nem működött, de mivel általában nagyobb kincs az idő, mint a tárhely, a *zip* tömörítés is teljesen megfelelő.
A `partimage`-et lehet egy kiszolgáló közbeiktatásával is használni, de nálunk az iskolai gépeket ellátó szerveren nem *UHU Linux* van, hanem *Fedora*, az erre fordított `partimage` pedig valahogy nem volt kompatibilis az általam használttal. Pedig ez sok bosszúságon átsegített volna...
Így a helyi gépen készült képmás-fájlt egyszerűen felmásoltam a *Samba* szerverre, annak is a publikus részére, hogy később ne kelljen jelszót megadni (törölni vagy felülírni így sem tudják mások, ha jól van beállítva).

A nagy sokszorosítás

Az előkészületek után eltöprengtem azon, hogy pontosan hogyan is érdemes a képmás-fájlt sokszorosítani. Hasonló helyzetbe kerülhet egy netkávézó vagy egy teleház rendszergazdája, ahol szintén érdemes lehet bizonyos időközönként újratelepíteni a gépeken futó programokat. Nagy szerencse, ha ugyanolyan gépekről van szó, mint ebben az esetben.
Ami a merevlemezek felosztását illeti, rendszergazdánk már eleve meghagyott 20 GB-ot a *Linux* számára, de azért volt még mit tenni. Hogy gyorsabban haladjak, a nálam levő többféle linuxos boot-CD-ről indítgattam a gépeket a particionálás végett. A friss *SuliX*-szal annyiban meggyűlt a bajom, hogy – az első verzióval ellentétben – ezen most nincs `cfdisk`, csak sima `fdisk`, és ez csak akkor dolgozott pontosan, ha cilinderszámot adtam meg nem MB-ot. (Az `fdisk` másként számolja például a +5000MB-ot, mint a `cfdisk`, márpedig itt fontos, hogy *pontosan* ugyanolyan partíciók keletkezzenek).
Munka közben érdemes jegyzetelni, mert – különösen ha nem sorban haladunk – könnyen előfordulhat, hogy elfelejtjük, mivel mit is csináltunk pontosan, és akkor most

hol is van a hogyishívják. Bátrabbak próbálkozhatnak az `sfdisk`-kel, ami parancssorból tud particionálni (interaktív közreműködés nélkül). A később említendő autorun fájlba ezt is be lehet írni, bár én jobbnak láttam ezt a lépést óvatosabban, kézzel elvégezni.
A képmás-fájlok gépekre írása során két úton is haladhatunk. Akinek kevés gépet és nem túl gyakran kell klónoznia, annak megfelelő lehet a „gyári" ***SystemRescueCD*** is, aminek indulásakor be kell írni egy viszonylag hosszú paraméter sorozatot. Aki viszont szán egy kis energiát arra, hogy újraírja a telepítő CD-t a saját képmására szabott alapértelmezett paraméterekkel, annak a rászánt idő valószínűleg bőven megtérül a későbbiekben.
Nézzük először a „gyári" ***SystemRescueCD*** használatát. Még ebben az esetben is fontos szempont, hogy úgy töltsük be a gépeket, hogy ki lehessen venni munka közben belőle a CD-t, illetve automatikusan el lehessen érni egy/több elkészített parancsfájlt egy ***Samba*** szerverről betöltés után, amit a CD módosítása nélkül is módosítani lehet.
Erre a következő parancs szolgált segítségül (amit a bootolás előtt lehet megadni):

```
fb800 cdcache setkmap=18 ar_nowait autoruns=2,3,8
➥ar_source=//szervergepunk/pub
```

vagyis frame buffer üzemmód, 800x600 felbontással.
A `setkmap` (vagy `nokeymap`) paraméter megadásával kiküszöbölhetjük, hogy betöltés közben várjon a gép, míg megadjuk az általunk igényelt billentyűzet-kiosztást.
Az `ar_nowait` azt jelenti, hogy a rendszer az `autorun` után ne várjon az ENTER leütésére `autoruns`-ként megadott számokkal befolyásolhatjuk, hogy az `ar_source` könyvtárban elhelyezett `autorun1`, `autorun2`, `autorun3`... fájlok közül melyik fusson le. Ha nem írunk ilyen számsort, akkor csak az autorun fájlt fogja lefuttatni.
Szintén egy sokat átszenvedett tanulság, hogy az `autorun` parancsfájlok csak egysorosak lehetnek, viszont szabad pontosvesszőt használni a parancsok elválasztásához.
Itt a 2-es fájlba beírtam, hogy

```
umount /mnt/cdrom; eject
```

aminek hatására ki lehetett venni a cd-t a klónozás megkezdése előtt (és be lehetett tenni a következő gépbe – ennek jelentőségét csak az tudja, aki próbált már >=30 gépet felhúzni).
A 3-as fájlban felcsatoltatom a */opt*-ba (mert ott úgysincs semmi fontos a SystemRescueCD-n) a számomra szükséges Samba könyvtárat:

```
smbmount //szervergepunk/publ /opt -oguest
```

Itt az utolsó opció is kincset ér, mert elkerüli a jelszókérést.
Végül a 8-as fájl elindítja a `partimage`-t:

```
partimage restore -z1 -b -o /dev/hdaX
➥FELCSATOLT_SAMBA_KONYVTAR_AZ_IMAGE_FAJLLAL
```

Fontos, hogy a `restore` szó a kapcsolók előtt legyen.
Aki nem sajnálja a fáradságot, hogy saját igényeihez igazítsa a boot CD-t, az a következőképpen járhat el.
Másoljuk ***SystemRescueCD*** tartalmát egy új könyvtárba. Érdemes talán törölni a „felesleges" részeket (például a dokumentációt). Így egy bankkártya méretű CD-re is felfér a megmaradó adatmennyiség. Ezután keressük meg az ***isolinux.cfg*** fájlt, és írjuk át a `default` sort. Ezzel `default myconf` lett az eleje, amelynek tartalmát a következőképpen definiáljuk:

```
label myconf
        kernel vmlinuz1
        append initrd=initrd1 acpi=off root=/
➥dev/ram0 init=/linuxrc setkmap=18 vga=788
➥cdcache ar_nowait ar_source=//szervergepunk/pub
```

A lényeg itt is az `ar_source`-ban van, ami az `autorun` forráshelyére utal. Ez egy hálózati könyvtár, és az itt elhelyezett `autorun` fájlt (vagy akár többet) végrehajt a CD által futtatott ***Linux*** azon a gépen, ahol indítottuk**!** Ez igen elegáns és rugalmas lehetőség, valljuk meg.
Az `autorun` helye persze nemcsak ***Samba*** kiszolgáló lehet, hanem bármi: USB memória, merevlemez vagy akármilyen alkalmas adathordozó. Az itt található `autorun` parancsfájlba aztán be lehet „rámolni" mindazt, amit szeretnénk végrehajtatni. Ezek akár az image fájlban végrehajtott utólagos módosítások is lehetnek (például elírtuk a ***GRUB menu.lst*** egy sorát, visszamaradt egy ***postmaster.pid*** fájl, ami meggátolja a ***Postgresql*** indítását s emiatt törlendő stb.)
Nálunk a következő műveletsor futott:

```
umount /mnt/cdrom; eject; smbmount //mester/pub
➥/opt -oguest; partimage restore -z1 -b -o /dev/
➥hda6 /opt/img/vect6c; partimage restore -z1 -b
➥-o /dev/hda5 /opt/img/uhu5c; mkdir /root/5;
➥mount /dev/hda5 /root/5; perl -pi -w -e 's/
➥gfxmenu \(hd0,.\)/gfxmenu \(hd0,4\)/' /root/5/
➥boot/grub/menu.lst; grub-install --root-
➥directory=/root/5 /dev/hda; rm -f /root/r/var/
➥lib/postgres/data/postmaster.pid; umount -a;
➥reboot
```

Az elején, amint lehetett, kidobattam a CD-t, hogy tovább lehessen lépni a következő gépre. Két ilyen CD-vel már vígan lehet haladni egy nagyobb gépteremben is. Egy mai számítógépen 3 perc alatt van a `cdcache` végrehatása (azaz az ideiglenes fájlrendszer memóriába vitele) és az időigényes ***hotplug*** eszközök vizsgálata. (Ez utóbbi a klónozáskor felesleges, de nem sikerült megtalálnom, hogyan lehetne kiküszöbölni a futtatását. Érdeklődőbb hackerek negyedére csökkenthetik a ***SystemRescueCD*** betöltési idejét, ha ezt a lépést kivágják.)
Mint látható, egyszerre akár több partíciót is feltölthetünk. Példánkban az ***UHU*** mellett felmásoltam egy ***VectorLinux*** partíciót is (***vect6c***). Erről a disztribúcióról már volt szó a ***Linuxvilágban*** is**.** Kis gépen is jól fut, és nagyon friss a benne található kernel. Érdemes megismerkedni vele – hátha anyósunk régi gépére grafikus levelezőprogramot kell valaha telepítenünk, és akkor más nemigen jön szóba.
A saját CD elkészítésére vonatkozó információkat prózai módon úgy derítettem ki, hogy ***mc***-ből F3-mal ránéztem az iso fájl elejére. Ezt találtam:

```
mkisofs -J -R -l -o ../liveiso.iso -b
➥isolinux/isolinux.bin -c isolinux/boot.cat
➥-no-emul-boot -boot-load-size 4 -boot-info-table
➥-V Linuxvilag_63
```

A ***SystemRescueCD*** honlapján (***www.sysresccd.org/howto/sysresccd-howto-advanced-customization.php***) találunk eligazítást arra vonatkozóan is, hogy magát a kernelt

hogyan lehet újrafordítani, de erre nem volt szükségem.
A `partimage` tudja kezelni a *MBR*-t is (ez szintén betehető az *autorun* fájlba), de én megelégedtem egy *GAG* telepítésével, aminek az az előnye a *GRUB*-bal meg *LILO*-val szemben, hogy nem árt neki, ha alatta le-föl pakolgatjuk a különböző *Linuxokat*. Akkor derült ez ki, amikor egy korábbi *Linux* alatt átszabtam a partíciót, és a *GRUB* innentől kezdve nem látta a neki szánt *menu.lst* fájlt (és emiatt persze más operációs rendszereket se tudott betölteni).
A *GAG* telepítéséhez szükséges lépéseknek megfelelő néhány karaktert célszerű egy papírra felírnunk, mert így 30 másodperc alá lehet szorítani a telepítését, ha nem kell mindig megérteni és kitalálni a következő leütendő billentyűt).
A „legalább a *Windows* menjen, de az automatikusan” változat telepítéshez a következő gombokat kell leütni:

```
4-Enter-3-J-S-O-B-win-Enter-Enter-C-B-5-Enter-2-M-
➥Enter-V
```

majd újratöltés vagy kikapcsolás.
Ha mégis a *GRUB* mellett maradunk végleges megoldásként, akkor a klónozást követően (vagy akár `autorun`-fájlba írva, mint fentebb is látszott) érdemes kiadni a

```
grub-install --root-directory=
➥FELMOUNTOLT_FRISS_ROOT_KONYVTAR /dev/hda
```

parancsot, ami a *MBR*-ba írja a – remélhetőleg jól előkészített, és a frissen becsatolt fájlrendszer *boot/grub/* könyvtárában található *menu.lst* által meghatározott – GRUB-ot.
(A *chroot*-tal valamiért nem jártam sikerrel.) Itt érdemes megfontolni, hogy ha egy korábbi Linuxunknál más számú partícióról indult a *GRUB*, mint a legutóbbi telepítéskor, akkor nemcsak a `kernel` és az `initrd` szakaszokban levő partíciószámokat érdemes átírni, hanem a *menu.lst* elején levő

```
gfxmenu (hdX,Y)/boot/themes/uhu
```

sorban az X és Y értékeket is, ha azt szeretnénk, hogy megmaradjon a grafikus indítómenünk. Továbblépési lehetőséget jelenthetne, ha „multicast” módban sikerülne letölteni a képmás fájlt a szerverről, azaz minden adatcsomag minden géphez eljutna. Ez nyilván felgyorsítaná a klónozást.
A *Linux* sokszorosításával az egyik célom az volt, hogy megtapasztalhassák a diákok: a legtöbb feladat ugyanúgy megoldható nyílt alkalmazásokkal, mint zárt kódú változatokkal. Azt is ki akartam deríteni, hogy melyek azok az alkalmazások, amelyeket szeretnének használni, de pillanatnyilag nem hozzáférhetők, vagyis hogy a diákok szerint hol vannak a Linux gyenge pontjai. Eközben az az életérzés is sokat jelentett mindannyiunk számára, hogy egy olyan, szabadabb világ építőköveit csiszoljuk, ami nem a pénzre épül, hanem „ajándékozó társadalom” paradigmájára, ahol az a nagyobb, aki többet ad, többet tesz a közért.

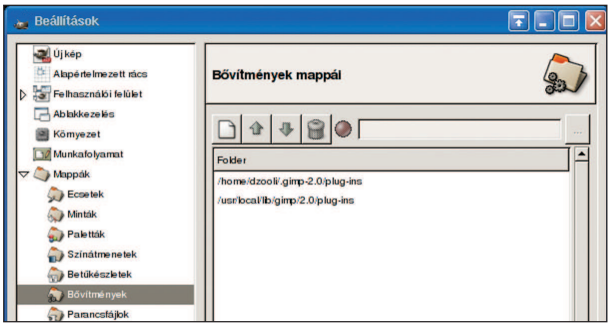
Szabó Zoltán
Három gyermekével és feleségével Pannonhalmán él. Tíz éve kísérletezik a Linuxszal. Matematikát és informatikát tanít, diákotthonban keseríti a rábízottak életét. Szívügye a PHP és a PostgreSQL. (szz@freemail.hu)

Bővítőmodulok segítséggel

A cikksorozat utolsó részében megvizsgáljuk annak lehetőségét, hogy hogyan írhatjuk át saját igényeinknek megfelelően a már elkészült GIMP bővítményeket.

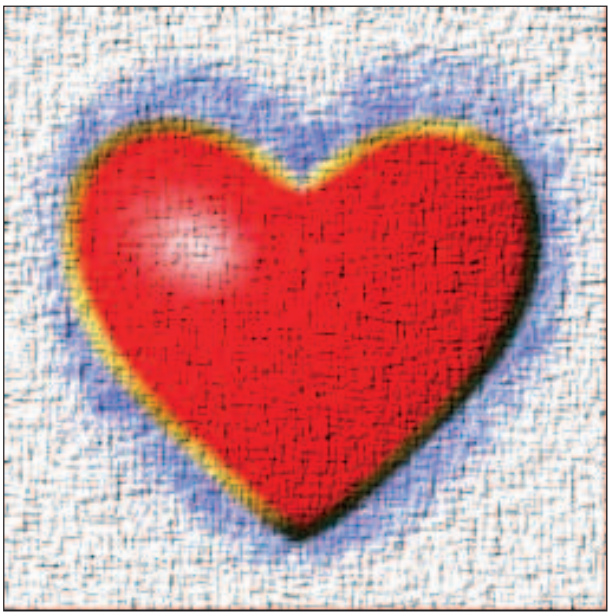
Tekintettel arra, hogy már néhány modul rendelkezésre áll ***Python*** nyelven is, így amennyiben megfelelő kiindulási alapot találunk, apróbb változtatásokkal könnyedén hozhatunk létre újabb hatásokat. A meglévő bővítményeket tároló könyvtár a ***GIMP*** eszköztár ablakában a ***File->Beállítások*** párbeszédablakban tekinthető meg. Válasszuk ki a bal oldali listában a ***Mappák->Bővítmények*** elemeket és nézzük meg a jobb oldali részen megjelenő adatokat. Ez a könyvtár az *1-es képen* látható esetben a ***/usr/local/lib/gimp/2.0/plug-ins***.
A következő lépés, hogy kiválasztjuk a számunkra megfelelő kiindulási alapot képező bővítményt. A ***GIMP***-ben hozzunk létre egy új képet vagy nyissunk meg egy már meglévőt és a ***Python*** menüben válasszunk egy modult. Nézzük meg, hogy melyiknek az eredménye áll legközelebb az elképzeléseinkhez. Az alábbi példában a ***Clothify*** modult egészítjük ki elképzeléseinknek megfelelően mégpedig úgy, hogy a mintázatot színes plazma aláfestéssel tesszük látványossá.
Tekintsük át, milyen változtatásokra van szükség a modul regisztrációs részében. Az első listán látható az eredeti ***Clothify*** modul bejegyzését végző kódrészlet.

```
register(
  "python_fu_clothify",
  "Make the specified layer look like it is
  ➥printed on cloth",
  "Make the specified layer look like it is
  ➥printed on cloth",
  "James Henstridge",
  "James Henstridge",
  "1997-1999",
  "<Image>/Python-Fu/Alchemy/_Clothify",
  "RGB*, GRAY*",
  [
    (PF_INT, "x_blur", "X Blur", 9),
    (PF_INT, "y_blur", "Y Blur", 9),
    (PF_INT, "azimuth", "Azimuth", 135),
    (PF_INT, "elevation", "elevation", 45),
    (PF_INT, "depth", "Depth", 3)
  ],
  [],
  python_clothify)
```



1. ábra A bővítményeket tároló könyvtár

Számunkra tulajdonképpen csak a párbeszédablak felépítéséhez szükséges változók definíciója érdekes, amelyet a (PF_ kezdetű sorokban határozott meg a bővítmény készítője.
A kiegészítésünkhöz még két változóra lesz szükség. Az egyik a plazma hatás örvénylésének mértékét adja majd meg, míg a másik a létrejövő rétegek átlátszatlanságát. Az átlátszatlansággal tudjuk majd befolyásolni, hogy milyen mértékben érvényesül a mintázat és mennyiben marad meg az eredeti hátteret képező kép. Az örvénylés meghatározására szolgáló változó neve legyen `turbulence`, típusa lebegőpontos. Az átlátszatlanság meghatározását egy csúszka segítségével végezzük, melynek értéke nullától százig változhat. A programban majd lebegőpontos értékként használjuk, de a bejegyzést végző kódrészletnél ennek nincs jelentősége. A változónk neve legyen `opacity`.
A második listán látható sorokat írjuk tehát az eredeti kódban a szögletes zárójel bezárása elé, de ne felejtsük el az eredetiben a sor végén vesszővel jelezni a ***Python*** értelmező számára a folytatást.
Látható az eredeti részletben, hogy a menüben hol helyezkedik majd el a bővítmény. Ezt a sort is javítsuk ki valamilyen más névre. Jelen esetben a ***Python-Fu->Alchemy->Plasma Clothify*** nevet választottam. A modul fő eljárásának a nevét is megváltoztattam, így a bejegyzést végző ***register*** eljárásban is az utolsó paraméterben a megváltozott nevet kell megadni. A 2-es listán látható a kódrészlet a leírt változtatások elvégzése után. Az első paraméterben megadott nevet is ki kell cserélnünk valamilyen más azonosítóra,



2. ábra Az eredeti bővítmény

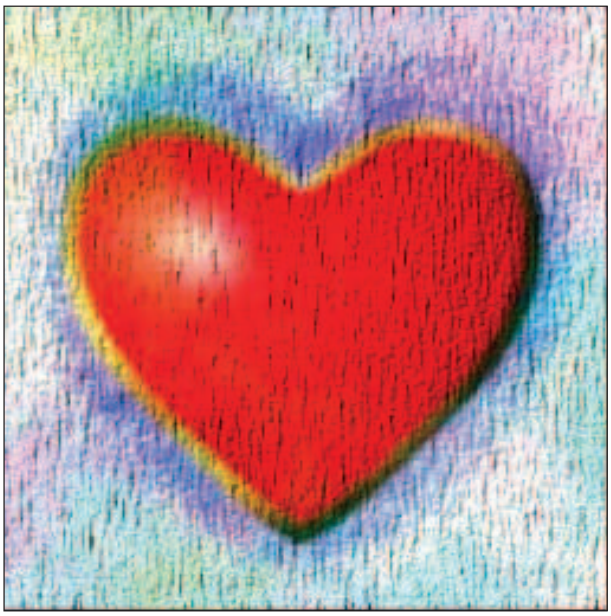
hiszen a *GIMP* ezen név alapján tárolja a saját bővítmény-adatbázisában az új eljárást. A bővítmény neve az új változatban `python_fu_plasmaclothify` lesz.

```
    # eredeti részlet
    (PF_INT, "elevation", "elevation", 45),
    (PF_INT, "depth", "Depth", 3),

    # örvénylő hatás beállítására
    (PF_FLOAT, "turbulence", "Plasma turbulence",
    ➥0.5),
    # rétegek átlátszatlansága (alapértelmezés 20%)
    (PF_SLIDER, "opacity", "Layers opacity", 20,
    ➥(0, 100, 1))
],
[],
python_pclothify)
```

A változók és a párbeszédablak beállítása után következhet az érdemi munka.
Lényegében a változtatás annyiból áll majd, hogy még egy plusz réteget adunk a képhez, melyen a plazma hatást létrehozzuk. A hatások elkészítése után a rétegeket a képre lapítjuk, így érjük el a végleges eredményt. Az eredeti bővítménnyel ellentétben most nem egy új képet hozunk létre hanem a *GIMP* által átadott képen dolgozunk. Az eredeti modulban minden műveletet az `img` változó által tárolt képen hajtunk végre. Első dolgunk tehát az legyen, hogy ahol hivatkozást találunk erre a változóra, azokon a helyeken javítsuk ki `timg` névre a megadott változót. Hogy miért pont `timg`-re? Ha jobban megvizsgáljuk a bővítmény fő eljárását, észre fogjuk venni, hogy az első paraméter neve ***timg***. Ez az a változó, melyben a GIMP átadja a modul részére a feldolgozásra szánt képet. Tehát a kívánt eredmény elérése érdekében nekünk a program által adott képen kell dolgoznunk, vagyis azon, melyet a `timg` változóban kapunk.

Ezután az alapvető változtatás után már tulajdonképpen nincsen sok dolgunk. A plazma hatást a *GIMP* bővítményei között találjuk, tehát az alábbi lépésekben összefoglalható minden szükséges változtatás. Elsősorban, miután az eredeti változat szerint elkészült a két ruhaszerű mintázatot megjelenítő réteg, következhet a harmadik réteg létrehozása. A harmadik rétegen jelenik meg a plazma hatás. Hozzunk létre egy új réteget a képen a `gimp.Layer()` objektum elkészítésével. Itt már a létrehozásnál adjuk meg a felhasználó által beállított átlátszatlanság értéket az `opacity` változó segítségével. Mivel a `gimp.Layer()` objektum létrehozásakor a hatodik paraméter az átlátszatlanság, így nincsen nehéz dolgunk. Az új réteg megjelenítési módja legyen `HARDLIGHT_MODE`, hogy jobban láthatóvá váljon a plazma mintázat a modul futtatása után. A példában használt réteget tároló változó a `layer_alma` nevet kapta. A réteg létrehozása után következhet a plazma hatás elkészítése. A *GIMP*-ben minden bővítményt vagy eljárást elérhetünk más bővítményekből is, tehát a nyugodtan meghívhatjuk a `pdb.plug_in_plasma()` eljárást. Az eljárás négy paramétert vár. Az első legyen a kép, melyhez az új réteg tartozik. Ez jelen esetben megegyezik a *GIMP* által átadott `timg` képpel. A második paraméter az a réteg, melyen a plazma hatás megjelenik. Példánkban ez a `layer_alma` réteg. A harmadik paraméter a véletlenszámok létrehozásához használandó alapérték. A bővítmény elején olvassuk be a ***Python*** rutinok között található `time` gyűjteményt, melyben megtalálható a `time()` függvény. Az éppen aktuális idő elegendő véletlenszerűséget biztosít a plazma hatás elkészítéséhez, tehát nyugodt szívvel használhatjuk az eljárás harmadik paramétereként. A negyedik szükséges paraméter határozza meg a hatás szabálytalanságát. Ez utóbbi értéket korábban örvénylésnek neveztük és a modul beállítására szolgáló ablakban a felhasználó állítja be.
Az új réteget természetesen az eredeti képhez is hozzá kell adni, erre szolgál a `timg.add_layer()` függvény.
Az utolsó lépésben már csak annyi a teendő, hogy a képen lévő rétegeket a `timg.flatten()` eljárás meghívásával egy réteggé „lapítjuk”. Végeredményben ez a pár lépés elegendő lenne a kívánt eredmény eléréséhez, azonban néhány apróbb módosításra még szükségünk van.
Elsősorban arról van szó, hogy az eredeti modulban a hozzáadott rétegek egyáltalán nem átlátszóak. Ez számunkra nem túl sikeres választás, hiszen ebben az esetben a háttérként szolgáló kiindulási képet már az első hozzáadott réteg teljesen el fogja takarni. A megoldást jelentő változtatás az első réteg létrehozására használt `gimp.Layer()` hívás jelenti. Ha megvizsgáljuk ennek az eljárásnak a paramétereit látható, hogy a hatodik paraméter – az átlátszatlanság mértéke –, állandó 100-as értéket kap. Ahogyan az általunk létrehozott rétegnél megoldottuk, itt is ki kell javítani ezt az állandót a felhasználó által beállított értékre. Cseréljük ki tehát a megadott értéket az `opacity` változóra.
A második rétegnél szintén hasonló problémába ütközhetünk, azonban nem találunk az első réteg létrehozásához hasonló hívást. Tulajdonképpen a második réteg az első másolataként jön létre, amit láthatunk a `layer_two = layer_one.copy()` sorban. A második rétegnél különféle változókat is beállítunk, mint például a `layer_two.mode = MULTIPLY_MODE` sorban. Itt állíthatjuk be az átlátszatlanságot



3. ábra Az új modul eredménye

is, arra azonban vigyázni kell, hogy a rétegnek lebegőpontos számot kell megadnunk. A `layer_two.opacity = float(opacity / 100)` sorban végezzük el ezt a beállítást. Utolsó lépés, hogy az első réteg megjelenítési módját (hetedik paraméter a `gimp.Layer()` hívásnál) szintén `MULTIPLY_MODE`-ként határozzuk meg, így jobban kiemeljük a végeredményben a réteg hatását.

Mivel csak a harmadik réteg elkészítése után van szükségünk a rétegek összevonására, az eredeti kódban szereplő `timg.flatten()` hívásokat megjegyzésbe tettem. Természetesen ki is lehet törölni ezeket a sorokat, de az érthetőség és a változtatások követése szempontjából érdemes a megjegyzést használni. Így később is tudjuk majd, milyen változtatásokat végeztünk a programban és nem veszítjük szem elől a bővítmény lényegét alkotó sorokat.

A harmadik listán látható a teljes bővítmény listája a korábban már tárgyalt bejegyzést elvégző `register` hívás nélkül. A második képen az eredeti modul által előállított képet tekinthetjük meg. A hármas képen az új változat által készített képet vehetjük szemügyre.

```
#!/usr/bin/env python

import time
import math
from gimpfu import *

def python_pclothify(timg, tdrawable, bx=9, by=9,
  azimuth=135, elevation=45, depth=3,
  turbulence=0.5, opacity=20):
     bx = 9 ; by = 9 ; azimuth = 135
     elevation = 45 ; depth = 3
     width = tdrawable.width
     height = tdrawable.height

# eredetileg új képen dolgozott a bővítmény.
# nekünk mindent az aktuális képen kell elvégezni
# img = gimp.Image(width, height, RGB)

  # atlatszóság beállítása a megadott
  # értékre és szorzás mód
     layer_one = gimp.Layer(timg, "X Dots", width,
      height, RGB_IMAGE,
      opacity, MULTIPLY_MODE)
     timg.disable_undo()
     pdb.gimp_edit_fill(layer_one, BACKGROUND_FILL)
     timg.add_layer(layer_one, 0)
     pdb.plug_in_noisify(timg, layer_one,
        0, 0.7, 0.7, 0.7, 0.7)
     layer_two = layer_one.copy()
     layer_two.mode = MULTIPLY_MODE

     # beállítjuk az átlátszóságot
     layer_two.opacity = float(opacity / 100)

     layer_two.name = "Y Dots"
     timg.add_layer(layer_two, 1)
     pdb.plug_in_gauss_rle(timg, layer_one, bx,
     ➥1, 0)
     pdb.plug_in_gauss_rle(timg, layer_two, by,
     ➥0, 1)
#     erre nincs szükség
#     img.flatten()
     bump_layer = timg.active_layer
     pdb.plug_in_c_astretch(timg, bump_layer)
     pdb.plug_in_noisify(timg, bump_layer,
        0, 0.2, 0.2, 0.2, 0.2)
     pdb.plug_in_bump_map(timg, tdrawable,
        bump_layer, azimuth,
        elevation, depth, 0, 0,
         0, 0, TRUE, FALSE, 0)

# új réteg és a plazma bővitmény használata
     layer_alma = gimp.Layer(timg, "Plasma",
        width, height, RGB_IMAGE,
        opacity, HARDLIGHT_MODE)
     pdb.plug_in_plasma(timg, layer_alma,
        int(time.time()), turbulence)
# réteget hozzáadjuk a képhez
     timg.add_layer(layer_alma, 1)
  # kép végső lapítása
     timg.flatten()

# az új képet nem használtuk,
# így törölni sem kell
#     gimp.delete(img)
```

Gyakorlatilag ez utóbbi tíz hónap alatt megtárgyaltunk minden olyan fontosabb dolgot, amire a *GIMP*-ben lehetőségünk adódik, így a sorozatnak nincsen hivatalos folytatása. Természetesen otthon vagy munkahelyén mindenki tovább folytathatja az ismeretek alkalmazását és bővítését, ebben a továbbiakban elektronikus levelezés útján a szerző is szívesen nyújt segítséget.

Fábián Zoltán

Védekezés levélkiszolgáló túlterhelése ellen

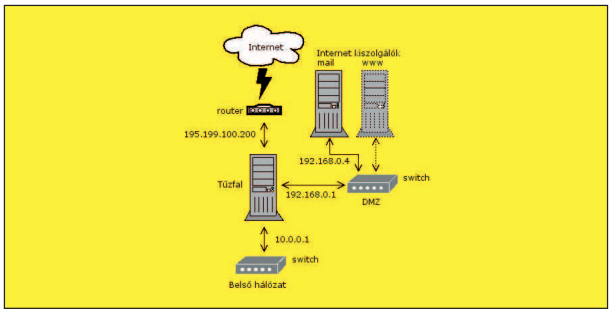
Egy győri szakközépiskola tanár-rendszergazdájaként linuxos kiszolgálókat üzemeltetek. A múlt év végétől, egyre gyakrabban tapasztaltuk azt a hibajelenséget, hogy a munkaállomásokon működő ügyfélprogramok hosszasan próbálkoztak egy-egy e-mail elküldésével, majd „Kapcsolat megszakadt” hibaüzenettel leálltak...

Mivel eddig nem tapasztaltam ilyen rendellenességet, több dologra is gyanakodtam. Végül levélkiszolgálónk forgalmának és terheltségének elemzésével világossá vált a valódi ok, és hamarosan a megoldást is sikerült megtalálnom.
A levelező kiszolgálónkon ***Debian Linux*** alatt ***Postfix MTA*** üzemel, amely a ***/var/log/mail.log*** fájlba naplóz. A naplóbejegyzéseket root-ként egyszerűen a ***Midnight Commander*** segítségével néztem meg. Itt csak 4 jellegzetes sort mutatok meg, amelyek egyetlen küldési kísérlethez tartoznak. Kiderült, hogy kiszolgálónkhoz gyors egymásutánban több IP-ről nagyszámú e-mail érkezett, mégpedig ismeretlen felhasználók címére. Ahogy a 3. sorban látható, az ismeretlen felhasználónak küldött levél a 450-es számú `User unknown` hibaüzenetet váltja ki és a bejegyzésbe a `reject` (elutasítva) kifejezés kerül:

```
1: Feb  6 06:51:22 mail postfix/smtpd[17399]:
➥connect from
➥mx6a.uniserve.ca[216.113.192.92]
2: Feb  6 06:51:23 mail postfix/smtpd[17399]:
➥06616662CA:
➥client=mx6a.uniserve.ca[216.113.192.92]
3: Feb  6 06:51:23 mail postfix/smtpd[17399]:
➥reject: RCPT from
➥mx6a.uniserve.ca[216.113.192.92]: 450
➥<estela.rudolph68@pattantyus-gyor.sulinet.hu>:
➥User unknown; from=<>
➥to=<estela.rudolph68@pattantyus-gyor.sulinet.hu>
4: Feb  6 06:51:28 mail postfix/smtpd[17399]:
➥disconnect from mx6a.uniserve.ca[216.113.192.92]
```

A ***Postfix*** csak akkor utasítja el az ismeretlen felhasználóknak küldött leveleket, ha megfelelően van beállítva. Ehhez az ***/etc/postfix/main.cf*** fájlban, a következő soroknak kell szerepelni:

```
1: alias_maps = hash:/etc/aliases
2: alias_database = hash:/etc/aliases
3: local_recipient_maps = $alias_maps
➥nis:passwd.byname
```



1. ábra Az iskola hálózatának felépítése

Az 1. és 2. sor az ***alias fájlok*** helyét adja meg. Ha például valakinek `horvathl` a felhasználói neve, de a `horvath.laszlo@domain` címmel szeretne levelezni, akkor ezt az alias fájlban kell megadni. Azt is itt lehet megadni, hogy a `titkarsag@domain` címre küldött e-mail melyik felhasználónak, vagy felhasználóknak a postafiókjába kerüljön.
A 3. sor arra utasítja a ***Postfixet***, hogy csak azon felhasználók címére fogadjon el e-mailt, akik vagy szerepelnek az alias-ban, vagy a ***NIS*** adatbázisban léteznek (a kiszolgálóinkon jelenleg ***NIS*** azonosítás működik).
Az elutasításnak természetesen más okai is lehetnek. Ha például a küldő ***IP***-nek nincs ***DNS***-bejegyzése, vagy ha továbbítónak (relay) akarja használni a kiszolgálónkat. Ilyenkor persze másféle hibaüzenetek keletkeznek.
Először arra gyanakodtam, hogy a rengeteg téves próbálkozás valamilyen e-mail küldő vírus tevékenységének a nyoma. A következő módon megvizsgált levelek tartalma azonban azt bizonyította, hogy valójában reklámküldési kísérletekről van szó. A ***/etc/postfix/main.cf*** fájlon a következő változásokat hajtottam végre:

```
1: #local_recipient_maps = $alias_maps
➥nis:passwd.byname
2: luser_relay =
➥felhasznalonev@pattantyus-gyor.sulinet.hu
```

Az 1. sor a postafiók meglétének ellenőrzését kikapcsolja (megjegyzésjelet tettem a sor elejére). A 2. sor hatására

a rendszer az ismeretlen felhasználóknak küldött leveleket az itt megadott postafiókba továbbítja.
A módosítás után a Postfixet újra kell indítani:
`/etc/init.d/postfix restart`

Vigyázat!!! Ha valahova olyan sok reklámlevél érkezik mint a mi címünkre, rövid idő alatt is rengeteg e-mail kerülhet a megadott fiókba. Ezért néhány perc után a */etc/postfix/main.cf* fájlt helyreállítva a ***Postfixet*** újra kell indítani.

A kiszolgáló forgalmi statisztikája és terheltségének vizsgálata

A napló tartalmának megismerése után kíváncsi voltam a szerver forgalmára. A ***Postfix*** esetében ehhez csak a ***pflogsumm*** csomagot kell telepíteni. Statisztika készítése a naplófájlban lévő valamennyi nap adataival:

```
pflogsumm.pl /var/log/mail.log > /root/statisztika
```

Csak az aktuális napi forgalom statisztikája:

```
pflogsumm.pl -d today /var/log/mail.log >
➥/root/statisztika
```

A `today` helyett `yesterday`-t írva pedig az előző napi forgalom statisztikáját készíti el.
Az elkészített statisztikából látszik, hogy egy nap alatt 40483 e-mailt fogadott (`received`), s ebből mindössze 591-et továbbított (`delivered`), 40028-at pedig elutasított (`rejected`):

```
Postfix log summaries for Feb 6

Grand Totals

messages

  40483   received
    591   delivered
      1   forwarded
     11   deferred  (33  deferrals)
      0   bounced
  40028   rejected
```

Lejjebb látható, hogy miért, illetve melyik küldőtől hány e-mailt utasított el a rendszer. Ebből az derült ki, hogy az elutasítások túlnyomó többsége azért történt, mert ismeretlen felhasználónak (`User unknown`) címezték a levelet.

```
message reject detail
---------------------
  RCPT
    Sender address rejected: Access denied
          26   spamblocker-
               ➥challenge@bounce.earthlink.net
          14   molnarlouis@bellsouth.net
          13   ErnoHoka@aol.com
    Sender address rejected: Domain not found
          20   AntiVirus@axxis.local
          14   Symantec_Mail_Security_for_
               ➥SMTP@MSL.COM
           3   MAILER-DAEMON@gcs.gateway
           1   admin@system.mail
    User unknown
         437   cox.net
         256   rr.com
         244   swip.net
         199   siteprotect.com
         154   netvigator.com
        .
        .
```

Ha a levelezőkiszolgáló elutasítja a nemkívánatos e-maileket, akkor miért terhelődik túl? A ***Postfix*** az internetről és a belső hálózatról is, az ***smtpd*** programjával fogadja az e-maileket, amely alapértelmezés szerint egyszerre legfeljebb 50 kapcsolatot tud kezelni. A ***Postfix*** 2.2-es változata óta lehetőség van ennek az értéknek a megemelésére, de a hardver gyorsasága és főleg a sávszélesség által szabott korlátok miatt nem biztos hogy ez célszerű.
Az adott pillanatban élő ***smtpd***-kapcsolatok számát a következő paranccsal néztem meg:

```
ps ax | grep smtpd | wc -l
```

Ennek értéke a fent bemutatott terheltség mellett legtöbbször 50 volt így nem tudott új kapcsolatot felépíteni a ***Postfix***. A levelező ügyfelek meghatározott ideig próbálkoztak, majd időtúllépés miatt írták ki a *„Kapcsolat megszakadt”* hibaüzenetet.

A bejövő smtp kapcsolatok időben történő korlátozása

Az első ötletem az volt, hogy csomagszűrő segítségével korlátozom az internet felől felépíthető kapcsolatok számát a levélkiszolgáló felé. Mivel intézményünkben a levelező kiszolgáló tűzfal mögött működik, a tűzfalgép szűrési szabályaiba építettem be a korlátozást:

```
1:  eth_kulso='eth0'
2:  IPTABLES=/sbin/iptables

4:  $IPTABLES -t nat -N pre_smtp
5:  $IPTABLES -t nat -A pre_smtp -m limit --limit
    ➥20/minute -j RETURN
6:  $IPTABLES -t nat -A pre_smtp -j DROP
7:  $IPTABLES -t nat -A PREROUTING  -i $eth_kulso
    ➥-d 195.199.100.200 -p tcp --dport smtp --syn
    ➥-j pre_smtp

9:   $IPTABLES -t nat -A PREROUTING -p tcp -d
     ➥195.199.100.200 --dport smtp -j DNAT --to
     ➥192.168.0.4:25
10:  $IPTABLES -t nat -A POSTROUTING -p tcp -s
     ➥192.168.0.4 --sport smtp -j SNAT --to
     ➥195.199.100.200:25
```

Itt a tűzfal külső (internetre csatlakozó) hálózati kártyájának IP címe `195.199.100.200`, míg a levélkiszolgáló címe `192.168.0.4`.

A beállítások jelentése a következő:

- 1. sor: A tűzfalgép külső (internetre csatlakozó) ethernet kártyájának megadása.
- 2. sor: Az `iptables` parancs elérési útja, hogy később ne kelljen mindig megadni.

- 4. sor: Új `pre_smtp` nevű lánc létrehozása
- 5. sor: Ha a kapcsolatok száma nem több percenként 20-nál, akkor visszatér a láncból, vagyis elfogadja (az értéket kísérletileg kell megállapítani, a szerver feldolgozóképességétől függ)
- 6. sor: Ha az előbbi sorban megadottnál több kapcsolat akar felépülni, akkor eldobja azokat.
- 7. sor: Ha a külső kártyán keresztül (internet felől) a tűzfalgép külső IP címére a TCP protokollon a 25-ös (smtp) portra kapcsolatteremtő csomag érkezik, ugrás a létrehozott `pre_smtp` láncra
- 9. sor: Ha TCP protokollon a tűzfal gép külső címén a 25-ös (smtp) portra érkezik csomag, átirányítja a levelező szerverre.
- 10. sor: Ha TCP protokollon a levelező szerver 25-ös (smtp) portjáról érkezik csomag, átirányítja a tűzfal gép külső kártyájára.

A fenti pár soros szkriptet természetesen módosítani kell, ha a levélkiszolgáló közvetlenül csatlakozik az internetre. Ekkor a `-t nat` részeket el kell hagyni, a `PREROUTING` bejegyzést `INPUT`-ra kell cserélni, a 9. és 10. sorok pedig természetesen elmaradnak.
A fenti korlátozás csökkentette a terhelést, a belső levelezést zavartalanná tette, de sajnos a kívülről érkező legális leveleket is várakozásra kényszerítette és időtúllépés miatt nem mindegyik jutott el a címzetthez. Olyan megoldást kellett ezért keresnem, ami lehetőleg csak a kéretlen levelek küldőit korlátozza.

Az „ellenséges” IP címek szelektív tiltása

Ötletem a következő volt. A levélkiszolgáló naplóállományát periodikusan átvizsgálva, ki lehet keresni azokat a címeket, melyekről megadott számú elutasított e-mail érkezik egy megadott idő alatt és ezeket egy szintén megadott időre ki kell tiltani. Természetesen nem lehet örök időre kitiltani egyetlen IP címet sem és nem is érdemes, hiszen az IP-k változnak. Azt is figyelembe kellett vennem, hogy olyan címek ne legyenek kitiltva, ahonnan hivatalos levél is érkezhet.
A kitiltásokat és engedélyezéseket naplózni kell, hogy bármikor meg lehessen nézni a tiltásokat és engedélyezéseket.
A feladat megoldására héjprogramot készítettem, mely a tiltást csomagszűréssel valósítja meg. Gondolom néhányan csóválják a fejüket, miért nem Perl-ben vagy esetleg C-ben írtam. Eredetileg az egyszerűség miatt döntöttem a héjprogram mellett, a gyakorlat azonban bebizonyította, hogy a héj és a különböző szűrőparancsok minden szükséges eszközt biztosítanak egy ilyen feladat hatékony megoldásához. Itt csak az érdekesebb sorokat mutatom be. A teljes program letölthető a ***Linuxvilág magazin*** webhelyéről (***http://www.linuxvilag.hu***).

```
1: #!/bin/sh
5: # Keszitette: Jaszberenyi Jozsef, 2005

21: MAX_REJECT=10
24: TILTOTT_ORAK=36
27: MAX_LOG=5000

30: LOGFAJL="/var/log/mail.log"
33: TILTOTTIPDIR="/var/log/ipdisable"
36: TILTOTTIPFAJL="tiltottIP.log"
39: NAPLO="naplo.log"
42: NEMTILTHATOIP="nemtilthatoIP"
45: TILTOTTIPFAJL="$TILTOTTIPDIR/$TILTOTTIPFAJL"
46: NAPLO="$TILTOTTIPDIR/$NAPLO"
47: NEMTILTHATOIP="$TILTOTTIPDIR/$NEMTILTHATOIP"
        .
50: ECHO=/bin/echo
51: [ -x $ECHO ] || exit 5
        .
        .
126: trap '$RM -f /tmp/$$_?.tmp; exit 2' 1 2 3 9 15

129: TMP1="/tmp/$$_1.tmp"
130: TMP2="/tmp/$$_2.tmp"
131: TMP_DOMAIN="/tmp/$$_3.tmp"
132: TMP_IPCIM="/tmp/$$_4.tmp"
133: TMP_TILTANDO="/tmp/$$_5.tmp"

137: if ! [ -f $NEMTILTHATOIP ]
138: then
139:   $ECHO "Nem talalom a $NEMTILTHATOIP fajl-t!"
140:   exit 5
141: fi

153: if ! [ -f $TILTOTTIPFAJL ]
154: then
155:     $TOUCH $TILTOTTIPFAJL
156: fi

162: HONAP=`$DATE +%b`
163: HONAPOK=`$ECHO $HONAP | $CUT -c 1 | $TR a-z A-Z`
164: HONAPOK=$HONAPOK`$ECHO $HONAP | $CUT -c 2- | $TR
➥A-Z a-z`
167: NAP=`$DATE +%e | $TR -d ' ``
170: TILTASI_IDO=$[`$DATE +%s` / 3600]

177: NAPLOARCHIV=$NAPLO.tgz
180: if [ -f $NAPLO ]
181: then
183:   MAX_LOG=$[$MAX_LOG * 1024]
185:   FILEMERET=`$LS -l $NAPLO | $AWK ' {print $5} '`
187:   if [ $FILEMERET -ge $MAX_LOG ]
188:   then
190:     $RM -f $NAPLOARCHIV
192:     $TAR cfz $NAPLOARCHIV $NAPLO
194:     $RM -f $NAPLO
195:   fi
196: fi

201: $ECHO "#------------------------------" >> $NAPLO
202: $ECHO "Start: "`$DATE` >> $NAPLO

204: # Kigyujti az aktualis napon keletkezett,
➥"reject:" kifejezest tartalmazo sorokat 9. mezojet
207: $GREP "reject:" $LOGFAJL | $AWK -v H=$HONAPOK -v
➥N=$NAP '$1 ~ H && $2 ~ N {print $9}' > $TMP1

209: # Kigyujti a domain neveket egy kulon listaba
```

```
210: $AWK -F[ '{print $1}' $TMP1 > $TMP_DOMAIN

212: # Kigyujti az IP cimeket egy kulon listaba
215: $SED s/^.*\\[// < $TMP1 | $SED s/].*$// >
➥$TMP_IPCIM

217: # Elollitja az uj listat soronkent egy IP-cim,
➥majd szokozzel a domain nev
218: $PASTE $TMP_IPCIM $TMP_DOMAIN > $TMP1

220: # Eltavolitja a nem tilthato IP-ket
221: for cim in `$CAT $NEMTILTHATOIP`
222: do
223:     $GREP -v ^$cim $TMP1 > $TMP2
224:     $CP -f $TMP2 $TMP1
225: done

227: # Sorba rendezi a sorokat IP szerint es meg-
➥szamolja melyik IP hanyszor fordult elo a listaban
228: $SORT < $TMP1 | $UNIQ -c > $TMP2

230: # Kivalogatja azokat az IP-ket, melyekrol a meg-
➥adott szamot
231: #  meghalado elutasitas tortent es kiirja
➥a tiltando IP-ket tarolo fajlba
232: $AWK -v DARAB=$MAX_REJECT '$1>=DARAB {print
➥$2":"$3}' < $TMP2 > $TMP_TILTANDO

235: # Kitiltja a tiltando cimeket ha nem szerepelnek
➥meg a tiltottIP fajlban
236: #  es listazza a tiltottIP fajlba idobelyeggel es
➥domain nevvel egyutt.
239: for sor in `cat $TMP_TILTANDO`; do
240:      cim=`$ECHO $sor | $AWK -F: ' {print $1} ' `
241:      domain=`$ECHO $sor | $AWK -F: ' {print $2} ' `
242:      if ! $GREP -s $cim $TILTOTTIPFAJL >>
➥/dev/null 2>&1
243:      then
244:         $IPTABLES -I INPUT -i eth0 -p tcp --dport
➥smtp -s $cim/32 -j REJECT
245:    $ECHO -n $cim >> $TILTOTTIPFAJL
246:    $ECHO -n : >> $TILTOTTIPFAJL
247:    $ECHO -n $TILTASI_IDO >> $TILTOTTIPFAJL
248:    $ECHO -n : >> $TILTOTTIPFAJL
249:    $ECHO $domain  >> $TILTOTTIPFAJL
250:    $ECHO "Tiltva:   $cim    $domain" >> $NAPLO
251:     fi
252: done

255: # Ha van olyan IP a tiltottIP fajlban ami a meg-
➥adott idonel regebben lett tiltva,
256: # akkor torli a tiltast, torli a tiltottIP
➥fajlbol a bejegyzest es naplozza
258: for sor in `cat $TILTOTTIPFAJL`; do
259:      # Kulonvalasztja a mezoket
260:      cim=`$ECHO $sor | $AWK -F: ' {print $1} ' `
261:      ERTEK=`$ECHO $sor | $AWK -F: ' {print $2} ' `
262:      domain=`$ECHO $sor | $AWK -F: ' {print $3} ' `
263:
264:     if [ $ERTEK -le $[$TILTASI_IDO-$TILTOTT_ORAK] ]
265:     then
266:         $IPTABLES -D INPUT -i eth0 -p tcp --dport
➥smtp -s $cim/32 -j REJECT
267:            $GREP -v $sor $TILTOTTIPFAJL > $TMP1
268:    $MV -f $TMP1 $TILTOTTIPFAJL
269:    $ECHO "Engedelyezve: $cim $domain" >> $NAPLO
270:     fi
271: done

274: $RM -f /tmp/$$_?.tmp

276: #Script futas befejezesi idopontjanak naplozasa
277: $ECHO "End: "`$DATE` >> $NAPLO
278: $ECHO

280: exit 0
```

A programfájlt természetesen futtathatóvá kell tenni: `chmod 755 ipdisable`

Elemzés

- A 21. sorban adjuk meg, hány elutasított e-mail után tiltjuk ki a küldő IP címét. Ezt az értéket kísérletileg kell meghatározni a terheltség függvényében. A program alapértelmezésként csak az aznapi bejegyzéseket vizsgálja.
- A 24. sor tartalmazza, hogy hány órára lesz kitiltva az IP cím. Célszerű ezt az értéket 24-nél nagyobbra állítani, mert az elutasított e-mailek jelentős része nem levélkiszolgálón keresztül érkezik, hanem közvetlenül internetre csatlakozott gépekről, melyek IP címe általában 24 óra után változik meg (a szolgáltató bontja a kapcsolatot és új IP-t oszt ki)
- A 27. sorban a naplófájl legnagyobb méretét adjuk meg (KB-ban), míg a 30-47 sorok bizonyos elérési utakat és fájlneveket adnak meg.
- A `nemtilthatoIP` nevű fájlban (42. sor) kell megadni azokat az IP címeket, illetve tartományokat, melyeket a programnak soha nem szabad kitiltania. Itt mindenek előtt célszerű megadni a belső hálózat IP tartományát. A fájlban megadható teljes IP cím (`193.233.21.14`), vagy IP tartomány is (`10.0.`).
- Az 50-123 sorokban megadjuk bizonyos ***UNIX*** parancsok elérési útját, illetve ellenőrizzük meglétüket. A 126. sor felhasználói megszakítás, vagy a gép újraindulása esetén gondoskodik az átmeneti fájlok törléséről, amelyek helyét a 129-133 sorokban adjuk meg.
- A 137-156 sorokban egyes fájlok meglétét ellenőrizzük. A `nemtilthatoIP` fájlnak létezni kell a `TILTOTTIPDIR` változóban megadott könyvtárban, különben a program futása befejeződik. A többi fájlt a program létrehozza, ha még nem léteznek.
- 162-164 sorokban az aktuális dátum hónap nevét kérdezzük le, majd konvertáljuk ugyan olyan formátumra, mint ahogy a ***Postfix*** naplófájlban szerepel. A 167. sorban a dátum napját kérdezzük le és töröljük a felesleges szóközt, ami egyjegyű szám esetén szükséges.
- 170. sorban időbélyeget állítunk elő 1 órás felbontással. Az időbélyeg segítségével tudjuk megállapítani, mennyi idő óta van egy IP kitiltva.

- 177-196 sorokban a naplófájl maximális méretét korlátozzuk. Ha van naplófájl és a hossza eléri a megadott max. méretet, akkor tömörítjük, majd töröljük a már betömörített eredeti naplót. Ha volt már betömörített naplófájl, akkor azt töröljük.
- 201-202 sorokban kezdjük a naplózást, a program futás kezdeti időpontjának kiírásával.
- 207. sorban a ***Postfix*** naplófájlból kigyűjtjük egy átmeneti fájlba az aktuális napon keletkezett, `reject:` kifejezést tartalmazó soroknak a 9. mezőjét. Az átmeneti fájlban a domain után szögletes zárójelek között lesz az IP cím. Ennek a formátumnak a kezelése nem lenne egyszerű, ezért át kell alakítani.
- 210-218 sorokban olyan formátumú fájlt hozunk létre, melyet a későbbiekben könnyen tudunk feldolgozni. A fájl minden sorában egy ***IP*** címet írunk, majd szóközzel a hozzátartozó domaint. A 210. sorban létrehozzuk a domaineket tartalmazó ideiglenes fájlt. Mivel a domain után szóköz nélkül egy kezdő szögletes zárójel van, az `awk` programmal könnyen szét tudjuk választani az utána lévőktől. A 215. sorban az ***IP***-ket tartalmazó ideiglenes fájlt hozzuk létreolyan módon, hogy `sed` parancs segítségével töröljük a szögletes zárójelek előtti és utáni karaktereket, beleértve a zárójeleket is. A 218. sorban egyesítjük az ***IP***-ket és a domaineket tartalmazó fájlokat.
- 221-225 sorokban töröljük a fájlból a nem tiltható ***IP***-ket olyan módon, hogy csak azokat a sorokat másoljuk át, melyek nem úgy kezdődnek, mint a `nemtilthatoIP` sorai.
- 228. sorban megszámoljuk, hogy melyik sor hányszor fordul elő. A rendezésre azért van szűkség, mert az `uniq` csak így működik helyesen.
- 232. sorban kiválogatjuk azokat a sorokat, melyek legalább megadott darabszámban fordulnak elő. Ezeket az ***IP***-ket fogjuk kitiltani, ha eddig nem lettek (nem szerepelnek még a `tiltottIP.log` fájlban).
- 239-252 sorokban tiltjuk ki az „ellenséges” ***IP***-ket: ehhez sorra vesszük a kitiltandó ***IP***-ket és ha még nincsenek kitiltva, akkor kitiltjuk azokat. A tiltást naplózzuk a `tiltottIP.log` fájlba „IP:időbélyeg:domain” formában, valamint a `naplo.log` fájlba. A 242. sor végére azért írtam a `>> /dev/null 2>&1` részt, mert különben a `grep` kimenetéről minden futás után e-mailt küld.
- 258-271 sorokban engedélyezzük azokat az IP-ket, melyek már elég ideig voltak tiltva. Az engedélyezett IP bejegyzését töröljük a `tiltottIP.log` fájlból és az engedélyezést naplózzuk a `naplo.log` fájlba.
- 274. sorban töröljük az ideiglenes fájlokat.
- 277-278 sorokban fejezzük be a naplózást. A program futásának befejezési idejét a `naplo.log` fájlba írjuk.

Időzítés és a működés ellenőrzése

A programot rendszeres időközönként kell futtatni, mivel óránként jelentkeznek újabb kalóz IP-k. Ez legegyszerűbben a `crontab`-bal oldható meg. Egy tetszőleges nevű fájlba kell a `crontab` utasítást írni, majd rootként bejelentkezve `crontab fájlnév` paranccsal hozzáadni a root nevében végrehajtandó `crontab` listához. Nálam a program 15 percenként fut, így a `crontab` utasítás a következőképpen néz ki:

```
5-50/15 * * * * /eleresi_ut/ipdisable >/dev/null
2>&1
```

A `>/dev/null 2>&1` részre azért van szükség, hogy a program futásáról ne küldjön e-mailt.
A program minden egyes futásakor ír a naplóba, ezért legegyszerűbben a naplófájl (***/var/log/ipdisable/naplo.log***) tartalmának megjelenítésével lehet ellenőrizni a működését:

```
#-----------------------------------------------
Start: Tue Feb 15 16:02:01 CET 2005
Tiltva: 159.53.206.179  smtpext15.bankone.com
        .
        .
Engedelyezve: 162.33.130.251 unknown
        .
        .
End: Tue Feb 15 16:02:19 CET 2005
```

Ha nem történt sem tiltás sem engedélyezés, akkor csak a programfutás indulásának és befejezésének időpontja látszik. A naplót érdemes rendszeresen átnézni, és megvizsgálni, hogy nem tiltott-e ki olyan IP-t, ahonnan fontos levél is jöhet.
A ***tiltottIP.log*** fájlba az éppen kitiltott IP-k találhatók `IP:időbélyeg:tartomány` formában:

```
193.229.0.43:307876:fep34-0.kolumbus.fi
204.140.14.154:307876:kitty.warnerbros.com
64.243.89.147:307876:unknown
        .
        .
206.168.3.177:307876:unknown
```

Az `iptables` paranccsal is lehet ellenőrizni, hogy milyen IP címek vannak kitiltva:

```
iptables -L INPUT -n | grep REJECT
```

A következő parancs az éppen kitiltott IP-k számát írja ki:

```
iptables -L INPUT -n | grep REJECT | wc -l
```

Tapasztalat

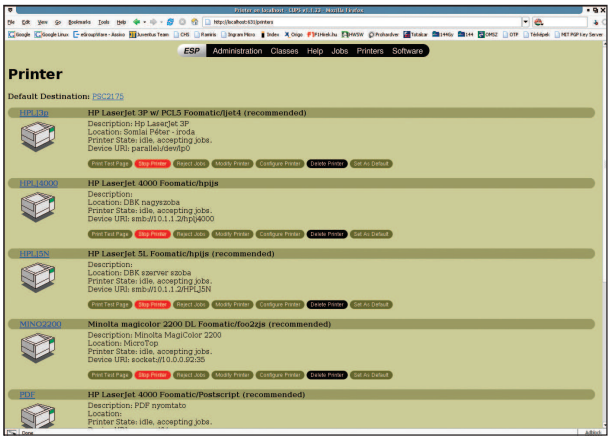
A program több hete működik. Az általunk használt gépen körülbelül 30 másodperc alatt lefut. Beüzemelése óta kollégáim nem panaszkodtak, hogy nem tudnak levelet küldeni, kézbesítetlen e-mail pedig csak az első napokban volt, amíg a `nemtilthatoIP` fájlban meg nem adtuk az összes szükséges helyet.

Jászberényi József
(jaszberenyij@pattantyus-gyor.sulinet.hu)
Szeret biciklizni, kirándulni, olvasni, sörözni és szabadban főzni. A stratégiai játékoktól a műszaki CAD programokig sok minden érdekli. Legtöbbet szerverprogramokkal foglalkozik és néha mérgelődik.

Samba – Windowsban is otthon (3. rész)

Lokalizáció, házirendek, nyomtatók.

Előző cikkem végére eljutottunk odáig, hogy egy működő *Windows* tartományt létrehoztunk, ehhez a már megismert módon felhasználókat társíthattunk, bejelentkezhettünk. A felhasználókhoz tartozó profilok a szerveren kerültek tárolásra – ezt nevezzük ***barangoló profilnak (roaming profile)*** -, így a hálózatunk bármelyik gépén bejelentkezve ugyanazt a felhasználói környezetet használhatjuk.
Ebben a cikkben teszek némi kiegészítést az előzőekben elmondottakhoz, beállítjuk a tartományi házirendet, létrehozzuk azt a futtatható állományt, amely minden bejelentkezéskor lefut a felhasználóknál, valamint nyomtatókat telepítünk. És a beállítások megkönnyítéséhez minden lépéshez mutatok működő példakódot is, amivel egy helyesen beállított rendszert kiegészítve pillanatok alatt fel tudjuk az új tudással ruházni a rendszert. Lássunk is neki...



1. ábra

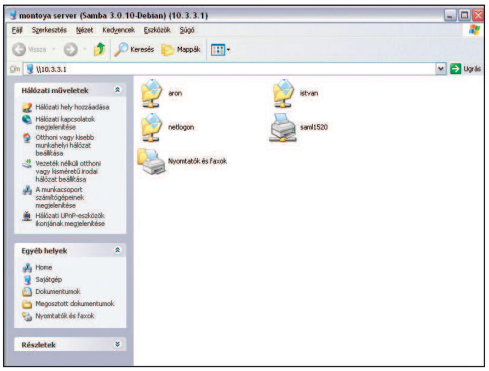
Lokalizáció, avagy a magyar nyelv szépségei

Ékezetes betűk, a hosszú ***Ő***, a hosszú ***Ű***. Minden rendszergazda és talán nem túlzás, minden tapasztaltabb felhasználó rémálma. Egy ékezetes karakter egy angol operációs rendszerben, egy nyugat-európai karakterkészlettel olyan gubancokat tud okozni, ahol a meg nem jelenített betű még a legkisebb problémák közé tartozik. Szélsőséges esetben az ilyen állományok olvashatatlanná, sőt extrém esetben akár törölhetetlenné is válnak. Egy rémálomba illő történetem van erről, amikor egy angol ***Windows NT4***-es rendszeren tárolt, de magyar ***Windows 98***-ak által létrehozott állományokat kellett egy új, magyar nyelvű ***Windows 2000*** alapú rendszerbe mozgatni. Természetesen az ***Ő*** és ***Ű*** betűk miatt a fájlokat közvetlen a két kiszolgáló között nem sikerült átmásolni, csak egy két rendszeren keresztülfuttatott kerülő útján sikerült az állományok nagy részét átmenteni.
Hasonló problémák sajnos a ***Samba*** kapcsán is könnyel felmerülhetnek, főleg az olyan heterogén rendszerekben, ahol az operációs rendszerek több nyelven is beszélnek, vagy olyan eltérő verziójú rendszereket üzemeltetünk, mint egy ***Windows XP*** és egy ***Windows 98***, vagy éppen egy ***Linux***.
Egyfelől kínosan oda kell figyelni arra, hogy a kliensek és a kiszolgálók azonos kódlapot használjanak, mert ezt figyelmen kívül hagyva komoly meglepetések érhetnek minket. Sajnos azt nem állítom, hogy ezeket az óvintézkedéseket megtéve nem fogunk ilyenbe belefutni, de legalább ennyit tegyünk meg és akkor kicsit nyugodtabban alhatunk.
„Melyik kódlapot válasszuk?" – merül fel a jogos kérdés.
Az abszolút megoldásnak a ***Unicode*** látszik, de sajnos jónéhány régi program inkább ***CP852***-t, vagy ***ISO-8859-2***-t használ, így tapasztalatom szerint érdemesebb inkább a korszerű ***Windows*** és ***Linux*** rendszereket is alapértelmezésként erre a kódlap családra visszaállítani. Persze az ideális megoldás az az, hogy ne használjunk sem a felhasználók nevében, jelszavakban, sem pedig a fájlnevekben ékezet betűket, de sajnos ez utóbbi elérése a felhasználóknál tapasztalatom szerint nem egy egyszerű dolog.
A sok problémafelvetés és boncolgatás után nézzük, hogy mit is kéne tenni, hogy a ***Sambát*** rávegyük egy adott kódlap használatára. A `[global]` paraméter szekcióba vegyük fel a `dos charset = 852`, illetve a `unix charset = iso8859-2` bejegyzéseket. Ezután a ***Samba*** a kapott karaktersorozatokat a közép-európai kódolás szerint fogja használni, amennyiben a kódolás a rendszeren telepített és értelmezhető.

A root, a rendszergazda és az administrator

Sokszor előfordulhat olyan helyzet, amikor egy adott felhasználó több néven jelenik meg egy rendszerben. Ennek talán legtöbbször előforduló esete, amikor a rendszerben az operációs rendszerek nyelve nem homogén, így tartalmaz például angol és magyar nyelvű operációs rendszereket is.
A rendszergazda felhasználói megnevezése ezen rendszerekben lehet ***Administrator***, lehet Rendszergazda, vagy egyéb más nyelvű elnevezés. Ahhoz, hogy ne kerüljünk bajba és a tartományunkban ezek a felhasználók ugyanazt a felhasználót jelentsék, ehhez egy felhasználónév össze-

rendelést kell készíteni. Ezt megtehetjük, ha az *smb.conf* állományban felvesszük a `username map = /etc/samba/usermap` bejegyzést. Ennek a változónak az értéke az az állomány, amelyben a felhasználói összerendelések szerepelnek.
Az elhelyezkedése gyakorlatilag tetszőleges, a lényeg, hogy az *smb.conf*-ban a helyes állományt adjuk meg. Jelen esetben én az */etc/samba/usermap* állományt használom, amelynek tartalma nálam a következő:



2. ábra

```
root = administrator rendszergazda
viktor = devel
```

A fenti állományt beillesztve a *Samba* konfigurációjába elértem azt, hogy amikor a tartományba a *root*, az *administrator*, vagy a rendszergazda felhasználóval lépek be, az gyakorlatilag ugyanazt a felhasználót jelenti, ugyanazt a profilt használom, ugyanaz a jelszavam, stb.. Ugyanígy a *viktor* és a *devel* felhasználók is azonos felhasználói fiókot jelentenek.

Bejelentkezési (logon) szkriptek és a házirendek – Nyomtatók kezelése

Mostanra meglehetősen sokat foglalkoztunk a fájlok, könyvtárak megosztásával, de alig esett szó a nyomtatókról, pedig egy irodai munkakörnyezetben ezek az eszközök, illetve ezen eszközök elérése legalább annyira fontos, mint az állományok elérése.
Mielőtt elmélyednénk a *Samba* és a nyomtatókezelés kapcsolatának rejtelmeiben vizsgáljuk meg egy kicsit, hogy miként is lehet *Linux* alatt nyomtatni.
Linux alatt szerencsére találunk jónéhány megoldást a nyomtatóink kezelésére, így választhatunk ízlésünknek megfelelően. Én is ezt tettem és a *CUPS* nyomtatórendszert választottam. A *CUPS* (azaz a *Common Unix Printing System*) egy olyan nyomtatórendszer, amely a *Linux* rendszerekben széleskörű támogatást élvez, könnyen konfigurálható és meglehetősen széles hardverskálát fed le. *Debian* alatt a *CUPS* csomagból telepíthető – a csomagot szerintem mindenki egész könnyen meg fogja találni – és a különböző nyomtatókhoz tartozó meghajtóprogramokat is telepíthetjük a kedvenc csomagkezelőnk segítségével.
A meghajtóprogramok egy nagy csoportja megtalálható a *hpijs* csomagban, illetve a *foomatic* csomagokban. Ezeket mindenképpen érdemes feltelepíteni. Ha valakinek olyan nyomtatója lenne, amely még ezekkel a csomagokkal sem működik, akkor érdemes a *www.linuxprinting.org* oldalra ellátogatni és onnan beszerezni a szükséges *ppd* fájlt. Ez a *ppd* fájl tartalmazza a nyomtató egyfajta *PostScript* leírását, így *Linux* alatt ez szükséges a nyomtató kezeléséhez.
Vannak már gyártók, akik gondolván a *Linux* felhasználókra már a gyári telepítőlemezen mellékelik ezt a pár kilobájtos állományt is – dicsérjük meg őket.
Ha eljutottunk oda, hogy feltelepítettük a *CUPS*-ot és a meghajtóprogramokat is, akkor állítsuk be a nyomtatónkat. Erre vannak natív kliensek a különböző *Linux* terjesztésekben, de használhatjuk akár a *CUPS* webes interfészét is, amit az adott gép 631-es kapuján tudunk elérni, valahogy így: ***http://10.0.0.1:631***
Lépjünk be a weblapra, és alapértelmezésként a root felhasználóval telepíthetjük is a nyomtatókat. Amennyiben nem csak a helyi gépről (localhost), illetve nem csak a root felhasználóval szeretnénk elérni a rendszert, akkor ezeket a beállításokat megváltoztathatjuk az */etc/cups/cupsd.conf* állományban.
Most viszont lépjünk tovább és telepítsünk fel egy nyomtatót. Nyissuk meg a *CUPS* weboldalát és lépjünk a *Manage printers* linkre, majd nyomjuk meg az *Add printer* gombot. A bejövő oldalon adjunk egy nevet – amely a régi programokkal való kompatibilitás érdekében ne legyen nyolc karakternél hosszabb, valamint ne tartalmazzon ékezetes és speciális karaktereket –, majd adjunk egy pontosabb leírást és egy elhelyezkedést a nyomtatóhoz.
A következő oldalon ki kell választanunk, hogy milyen módon szeretnénk a nyomtatóhoz csatlakozni. A *HP* hálózati nyomtatóinál elterjedt a *JetDirect*-en keresztüli csatlakozás, amelynek a *CUPS*-ban nagyon jó támogatása van, így ha ilyen nyomtatónk van, bátran használjuk ezt a megoldást. Amennyiben olyan nyomtatóhoz szeretnénk csatlakozni, amely egy másik *Unix/Linux* kiszolgálóhoz van csatolva és az támogatja az *ipp*, vagy *http* protokollon keresztüli nyomtatást – mint például a *CUPS* is –, akkor válasszuk ezt a csatlakozási módot. Ezzel megtehetjük azt, hogy egy másik *Linux* szerverre lokálisan, vagy hálózatban csatolt nyomtató erőforrásait használjuk anélkül, hogy a másik gépen különösebb beállításokat kéne tenni, vagy meg kéne a nyomtatót osztani. (Természetesen a *CUPS* beállításai között engedélyezni kell az *ipp* és *http* protokollon keresztüli elérést, valamint az sem árt, ha a tűzfalak nem tiltják a kapcsolatot, de más beállításra ezek után tényleg nincs szükség.)
A *CUPS* természetesen támogatja a lokális csatlakozást is, így választhatjuk a párhuzamos kapun való csatolást éppúgy, mint az *USB*-n való kommunikációt. És ezzel még mindig nem értünk a kapcsolódási lehetőségek tárházának a végére, ugyanis használhatunk akár *Windows* kiszolgálón, vagy másik *Samba* szerveren megosztott nyomtatós is, a *Windows Printer* illesztést használva.
Következő lépésként meg kell adnunk a pontos elérési utat a nyomtatóhoz, mégpedig egy szabványos internetcím formájában, tehát a protokoll megnevezése, kiszolgáló címe, erőforrás elérési útja formájában. (például: *ipp://my.server.net/printers/myprinter*). Természetesen a cím formátumának meg kell felelnie az előző lépésben kiválasztott eszköz típusának.
Ha megadtuk az eszköz elérési útját, akkor nincs más hátra, mint kiválasztani a nyomtató pontos típusát.
Itt viszont álljunk meg egy szóra. Mivel jelenleg *Windows* munkaállomásokhoz készítünk nyomtatókiszolgálót, ezért érdemes megfontolni, hogy az alábbi két konfiguráció közül melyiket választjuk.

Az első beállítás szerint kiválasztjuk a nyomtatónk pontos típusát és bekonfiguráljuk, majd az így megosztott nyomtatóra nyomtatunk. Ekkor azonban elképzelhető, hogy a munkaállomástól kapott nyomtatási fájl és a nyomtató lokális beállításai összevesznek és ez egészen odáig vezethet, hogy a nyomtatásunk végeredménye egy értelmetlen karaktersorozat formájában jön ki a nyomtatóból. Ezt a megoldást – tehát a pontos nyomtatómeghajtó kiválasztását a *CUPS*-ban – akkor javasolják, ha az adott gépről indítjuk a *CUPS* rendszeren a nyomtatást, tehát például tökéletes megoldás egy *Linux* munkaállomás esetén.
A másik lehetőség, hogy *RAW* nyomtatónak telepítjük a nyomtatónkat, így a nyomtatószerver gyakorlatilag egy nyomtatási sorként fog funkcionálni, tehát a kliensektől az ott telepített meghajtóprogrammal lefordított utasítások bekerülnek a sorba és onnan érkezési sorban távoznak a nyomtató felé. Ez utóbbi megoldást javasolnám a *Samba* nyomtatókiszolgáló alkalmazása esetén, mivel így tudjuk a munkaállomásokon a natív *Windows* meghajtókat használni, így a nyomtató minden funkcióját ki tudjuk használni. Sőt, mindjárt beállítjuk, hogy a felhasználóhoz a megfelelő nyomtatómeghajtó is feltelepüljön anélkül, hogy erre neki oda kéne figyelni.

És mi legyen a Sambával?

Végeztünk tehát a nyomtató telepítésével, most már csak meg kell osztani a *Windows* ügyfelekkel, ami természetesen a *Sambán* keresztül történik, méghozzá meglepően egyszerű módon. Amennyiben a *CUPS* nyomtatórendszert használjuk, akkor ezt elég tudatni a *Sambával* és ettől a pillanattól kezdve a *Samba* automatikusan megosztja a nyomtatókat. A beállításokat természetesen az */etc/samba/smb.conf*-ban kell elvégezni, teendőnk annyit, hogy az alábbi két sort beszúrjuk a konfigurációs állományba:

```
printcap name = cups
printing = cups
```

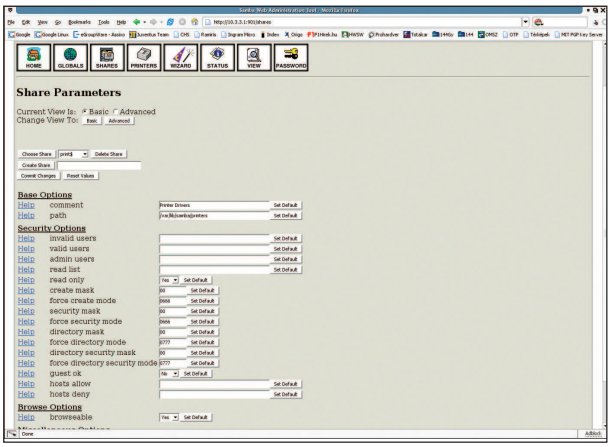
Ha ezt a két sort beszúrtuk az *smb.conf*-ba és abban szerepel az alapértelmezett nyomtatómegosztás is, akkor a nyomtatóink meg kell, hogy jelenjenek az adott gép kiosztásai között. A nyomtatók megosztására nálam az alapértelmezett beállítások vannak az *smb.conf*-ban, ezek a következők:

```
[printers]
        comment = All Printers
        path = /tmp
        create mask = 0700
        printable = Yes
        browseable = No
```

Amit a fenti beállításokon esetleg érdemes lehet megváltoztatni, az annak a könyvtárnak a helye, ahol a nyomtatás alatt az ideiglenes állományok tárolásra kerülnek. Akinek nem felel meg a */tmp*, az cserélje ki ízlése szerint.

És mi legyen a meghajtóprogramokkal?

Ha elkészültünk a kiosztással és csatlakozni is tudunk a nyomtatóhoz, akkor már csak egy dolgunk maradt, mégpedig a megfelelő meghajtóprogram telepítése a kliensre.



3. ábra

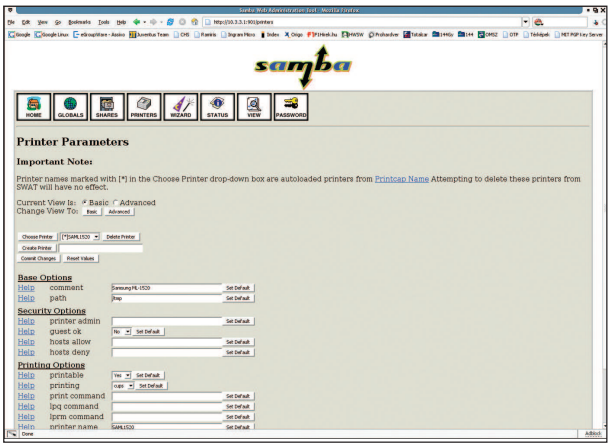
Ezt a lépést egész biztosan nem tudjuk kihagyni, hacsak az adott kliensre hasonló nyomtató már korábban nem került telepítésre, akkor a Windows egész biztosan szólni fog, hogy kérné a nyomtató meghajtóját. Ezt a lépést szintén lehet automatizálni, méghozzá úgy, hogy a kiszolgálón egy speciálisan erre fenntartott megosztásban kitesszük a nyomtatókhoz tartozó meghajtóprogramokat is, így a nyomtató csatolásakor azok automatikusan települnek a kliensre. Ehhez először létre kell hozni a kiosztást, utána pedig nyomtatónként hozzá kell adni a megfelelő meghajtóprogramot.
A kiosztást az alábbi kódrészlet beszúrásával hozhatjuk létre az *smb.conf*-ban:

```
[print$]
        comment = Printer Drivers
        path = /var/lib/samba/printers
        write list = root, administrator
```

A megosztáshoz azt hiszem sok magyarázatot nem kell fűzni. A `path` változó megadja, hogy a kiszolgálón hol lesznek tárolva a meghajtóprogramok, a `write list` pedig megadja azokat a felhasználókat, akik írási joggal rendelkeznek a könyvtárra, tehát akik nyomtató meghajtókat tudnak hozzáadni a rendszerhez.
Ezután már csak hozzá kell adogatni a programokat a kiszolgálóhoz, amit úgy tehetünk meg, hogy egy tartományi rendszergazda jogokkal rendelkező felhasználóval belépünk a tartományi kliensek egyikén, hozzáadjuk a nyomtatónkat a klienshez, majd megnyitjuk annak tulajdonságait és a megosztások kezelésére szolgáló fülön a további illesztőprogramok hozzáadása gombra kattintva elkezdjük rendszerenként hozzáadogatni a megfelelő illesztőprogramokat. Amelyeket hozzáadtuk a rendszerhez, azok innentől kezdve automatikusan települnek a nyomtatóval együtt.

Nyomtatók csatolása bejelentkezéskor

Lehetőségünk van arra is, hogy a nyomtatók telepítését, megosztások becsatolását, sőt akár programok telepítését is elvégezzük a felhasználó beavatkozása nélkül egy adott kliensen a felhasználó bejelentkezésével egyidejűleg. Ehhez mindössze annyit kell tenni, hogy összeállítjuk a megfelelő bejelentkezési (logon) szkriptet és azt elhelyezzük a korábban létrehozott *netlogon* megosztásunkban.



4. ábra

Ahhoz, hogy nyomtatókat telepítsünk, az alábbi parancsokat kell itt elhelyezni:

```
rem Nyomtatok telepitese
rem KMDI2011 nyomtato
rundll32 printui.dll,PrintUIEntry /dn /n
➥"\\myserver\KMDI2011" /q
rundll32 printui.dll,PrintUIEntry /in /n
➥"\\myserver\KMDI2011" /q
rundll32 printui.dll,PrintUIEntry /y /n
➥"\\myserver\KMDI2011" /q
```

Ezzel a parancskészlettel eltávolítjuk a nyomtatót, amennyiben az már létezett a rendszerben – mivel nem akarunk egy nyomtatóból több azonos példányt tartani – és hozzáadjuk, illetve telepítjük a megfelelő illesztőprogramot.

PDF nyomtatása Windows alatt

Nézzünk most egy érdekes problémát, amelyre egy ***Linux*** kiszolgálóval egészen könnyen találhatunk megoldást. ***PDF*** állományba való nyomtatás ***Windows*** rendszerek alatt szintén támogatott, akárcsak ***Linux*** rendszereknél, azzal az aprócska különbséggel, hogy ezekért a megoldásokért általában elég komoly összegeket kell fizetni. Ellenben felmerül a kérdés, miért fizessünk, ha megoldható ingyen is? Ráadásul úgy, hogy a teljes rendszert saját ízlésünknek megfelelően tudjuk kialakítani.
Nézzünk egy megvalósítási lehetőséget röviden összefoglalva. Amit tennünk kell az annyi, hogy készítünk egy nyomtató megosztást, majd az adott megosztásra érkező nyomtatási parancsokat átfolyatjuk egy megfelelő szkripten, amely lefutása után generál egy ***PDF*** állományt a kimeneten. Pofonegyszerű, ugye?
Nézzük egy kicsit konkrétabban, mire is van szükségünk. Először is készítsünk el egy nyomtató megosztást, amely nem egy eszközt, hanem egy futtatható állományt fog meghívni a beérkezett nyomtatási paranccsal. Helyezzük el az alábbi sorokat az ***smb.conf*** állományban

```
[smbpdf]
        comment = PDF Generator
        path = /tmp
        printable = Yes
        print command = /usr/sbin/pdfprint %s %U
        ➥%G %m %I %H
```

A fenti sorok jól láthatóan annyiban különböznek a nyomtatóink megosztásánál használt beállításoktól, hogy itt található egy `print command` paraméter. Itt kell megadni azt a futtatható állományt, amelyikkel a beérkező ***PS*** állományból a ***PDF***-et elő szeretnénk állítani. Ez tulajdonképpen egy tetszőleges általunk készített szkript is lehet, amely után akár egy adott könyvtárban elhelyezni, akár e-mailben továbbítja a nyomtatott állomány ***PDF*** verzióját.
Álljon itt egy példa a ***pdfprint*** szkriptre:

```
#!/bin/bash
#convert to pdf
#$1 = spool file $2 = uid $3 = gid $4 =
➥machinename $5 = ip $6 = homedir
FILENAME=pdf-$2-`date +%d%m%H%M%S`.pdf
OUTPUTPATH=$6/pdfwriter
echo converting $1 to $FILENAME for $2 of machine
➥$4... $6>> $OUTPUTPATH/pdfcreate.log
/usr/bin/ps2pdf $1 $OUTPUTPATH/$FILENAME >>
➥$OUTPUTPATH/pdfcreate.log 2>>
➥$OUTPUTPATH/pdfcreate.log
echo conversion finished, removing $1 >>
➥$OUTPUTPATH/pdfcreate.log
rm $1
echo done, setting permissions and notifing user
➥>> $OUTPUTPATH/pdfcreate.log
chown $2:$3 $6/$FILENAME >>
➥$OUTPUTPATH/pdfcreate.log
chmod 700 $6/$FILENAME >>
➥$OUTPUTPATH/pdfcreate.log
echo your new PDF is called $FILENAME. | smbclient
➥-M $4 -I $5
echo All done. >> $OUTPUTPATH/pdfcreate.log
echo >> $OUTPUTPATH/pdfcreate.log
```

Ezzel a kóddal a bemenetre érkező nyomtatási parancsot a feladó felhasználó ***home*** könyvtárában létrehozandó ***pdfwriter*** könyvtárban fogjuk elhelyezni és a felhasználó nevével, valamint a készítés időpontjával lesz elnevezve. Ami még nagyon fontos, hogy kliens oldalon olyan nyomtató illesztőprogramot használjunk, amely hagyományos ***PS*** állományt állít elő, amely lehetőség szerint nem tartalmaz semmilyen egyéb gyártóspecifikus kiterjesztést. Ilyenek a ***HP Laserjet PS*** és ***Apple Laserwrite*** nyomtatók, így ezek valamelyikét használva jó eséllyel működő ***PDF*** generátort kapunk.
Ezzel mostani cikkem végére is értem, remélem sikerült olyan érdekes megoldásokat bemutatni, amelyeknek a mindennapokban is hasznát lehet venni. A sorozat következő részében folytatjuk kalandozásunkat a ***Samba*** világában.

Illés Viktor (viktor@ei.hu)
23 éves, a BME műszaki informatikus szakának hallgatója, mellette weblapokkal, linuxos és windowsos rendszerekkel foglalkozik. Szabadidejét legszívesebben a szabadban tölti, teniszezik és kerékpározik.

FreeBSD – a szomszéd vár (6. rész)

Kiszolgálók – ACL és kiterjesztett attribútum a fájlrendszerben.

FreeBSD – a legtöbb UNIX rendszerhez hasonlóan – a könyvtárak és az állományok eléréséhez megfelelő jogokat követel meg, ezekkel a jogokkal képesek a felhasználók az állományokhoz hozzáférni. Ez a jogrendszer alapvetően három részre osztotta a felhasználók körét: tulajdonos, csoporttag és mindenki más. A UNIX történelem első évtizedében elegendőnek bizonyult, azonban mostanság ezek a jogok néhol erősen szűkösek lettek. S ez súlyos hiányosság, ha kiszolgálóként üzemeltetünk egy ilyen rendszert.

A UNIX jogosultságok rendszere

A régi nagygépes rendszerben minden bit kihasználásra került, mivel az erőforrások szűkösnek bizonyultak, s ennek megfelelően kellett jogosultságokat osztogatni állományokhoz és könyvtárakhoz. A leginkább erőforráskímélő megoldás felé hajlottak a fejlesztők, így csak az olvasás, az írás, illetve a futtatás jogait rendelték hozzá a fájlokhoz, valamint meghatároztak egy tulajdonost és egy csoportot is. A teljes jogrendszer tárolására elég volt néhány bájt adatterület, és ez nagyon jó kompromisszum volt a fejlesztők részéről. Azt is mondhatnánk erre a kialakításra, hogy több a semminél, de mindenre nem elég, s idővel a felhasználható tárterület is hatalmasra duzzadt, így ez a jogrendszer egyre inkább szűkös lett. Nézzük egy egyszerű példát, ahol van néhány felhasználónk: `tomy`, `noobi`, `joe` és `franko`. Ők mind egy munkahelyen dolgoznak, s néhány munkát meg kell osztaniuk egymás között. Például `tomy` dolgozik egy áramkör tervezésén, amelybe `noobi` is belesegít, viszont `joe` meg sem nézheti ezt a munkát, mert egy titkos fejlesztése a cégnek, miközben `franko` felügyelheti a tervezést. Ugyanakkor `franko` egy új programot fejleszt, amelyet `joe` és `noobi` csak megnézhet, viszont `tomy` ki van zárva a fejlesztők köréből.

Név	Munka1	Munka2
tomy	rwx	---
noobi	rwx	r--
joe	---	r--
franko	r--	rwx

A legjobb megoldás egy táblázatba foglalni a hozzáférések körét, amelyből hamar kiolvashatjuk, hogy ki és milyen jogosultságokkal képes hozzáférni a megadott állományokhoz. Megpróbálhatjuk ezeket a jogokat hozzárendelni a két állományhoz (`munka1` és `munka2`), de csoportok nélkül ez nem fog menni. Triviális az a tény, hogy a két munkaállomány csak kettő tulajdonossal jöhet létre, és a többiek jogait már csak a csoportjogokkal befolyásolhatjuk. Az is hamar észrevehető, hogy van mindkét munka esetén legalább egy olyan felhasználó, akinek egyetlen jogot nem tudunk adni. A második munka esetén létre tudunk hozni csoportot, amelynek tagjai (`noobi` és `joe`) csak olvasás joggal rendelkeznek, és elő is állt ennek a megoldása:

```
$ ls -l munka2
-rwxr-----  1 franko  munka2_csoport 0 Feb 22 13:35
➥munka2
```

Az első munkaállomány azonban szinte megoldhatatlan feladatot jelent, mivel nem tudunk kettő tulajdonost meghatározni, vagy kettő (egy `rwx` és egy `r--` jogú) csoportot rendelni az állományhoz. Tulajdonképpen ez a „kis” probléma az oka annak, hogy ***UNIX*** (vagy ***UNIX***-szerű) rendszerek nem tudtak betörni a fájlszerverek piacára, s nem tudták máig megtörni a ***Novell NetWare*** vagy a ***Microsoft Windows Server*** vezető szerepét ezen a területen.

Az ACL bekapcsolása

A ***FreeBSD 5.x*** az ***UFS2*** fájlrendszeren támogatja az ***ACL*** használatát (a ***FreeBSD 4.x*** esetén új rendszermagot kell fordítanunk), így képesek leszünk megoldani a jogosultsággal kapcsolatos problémáinkat. A kompatibilitás okán megmaradtak a megszokott jogok is, amelyeket azonos módon tudunk kezelni. Az ***ACL*** azonban alapértelmezés szerint nem aktív, bekapcsolásához a ***/etc/fstab*** megfelelő sorában meg kell adnunk az ***acls*** tulajdonságot is:

```
/dev/ad0s1f     /usr    ufs     rw,acls         1 1
```

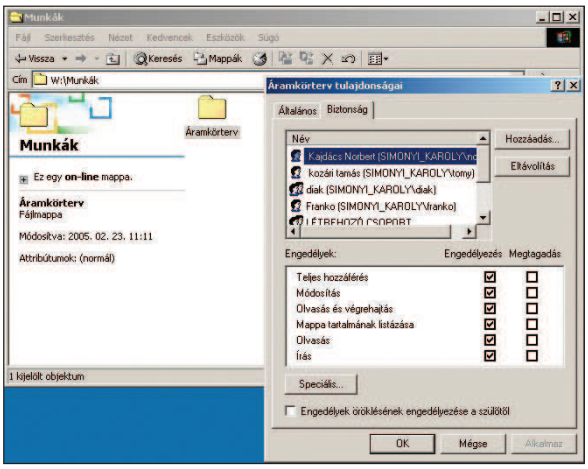
Természetesen bekapcsolhatjuk a `tunefs` programmal is, amelynél a `-a` kapcsoló `enabled` állásával tehetjük ezt meg:

```
$ tunefs -a enabled /dev/as0s1f
```

Ha mindezen túl vagyunk, akkor (elvileg) véget is ért a ***FreeBSD*** specifikus rész, s minden figyelmünket az ***ACL*** listák működésének megértésére tudjuk fordítani.

Hogy láthatjuk meg az ACL jogokat?

Kezdésképpen látni szeretnénk az ***ACL*** listákkal, hiszen – ha már beállítottuk – használnunk is kellene ezt a lehetőséget. Alapvetően a `getfacl` szolgál a hozzáférési listák lekérdezésére, de megvan a módja, hogy – például ***C*** nyelven – saját programból kezelhessük ezeket a listákat.



1. ábra ACL szerkesztése Windows alatt

Igazából bármelyik állományra vagy könyvtárra le tudjuk kérdezni a listát, egyszerűen kiadjuk a

```
$ getfacl munka2
```

parancsot, mire – ha minden jól megy – megkapjuk az állományunk eddigi beállításait:

```
#file:munka2
#owner:1001
#group:1008
user::rwx
group::r--
other::---
```

Ez tulajdonképpen azonos az

```
$ ls -l munka2
```

parancs által kiírt

```
-rwxr-----  1 franko  munka2_csoport 0 Feb 22 13:35
➥munka2
```

sorral, csak az *ACL*-listából nem tudjuk meg, hogy állománnyal vagy könyvtárral van-e dolgunk, illetve csak numerikusan láthatjuk a tulajdonost és a csoportját. A többi sor már az *ACL*-lista része, amelyet (angol nyelven) szövegesen olvasható formában láthatunk, s ennek a formátuma eléggé kötött, mivel csak a tradicionális *UNIX* jogokat láthatjuk. Tetszőleges állomány esetén találkozni fogunk *user*, *group* és *other* jogokkal, mint ahogy ezt meg is szokhattuk. A különbség annyi, hogy látunk kettőspontokat, amelyek elválasztják egymástól a kulcsszavakat és a jogok jelzéseit. A két kettőspont közé azonban olyan információkat tudunk írni, mint egy felhasználónév vagy egy csoport. Sőt, ezekből többet is felsorolhatunk; kezdésképpen oldjuk meg a `munka2` állomány jogait *ACL* használatával.

Hogy adjunk meg ACL jogokat?

A jogok kezelését a `setfacl` parancs végzi, ennek parancssorában meg kell adnunk azon jogot, amelyet hozzá szeretnénk fűzni az eddigi listához. Ehhez először is egy kis módosításra van szükség az állományunk általános jogai környékén:

```
$ chmod 600 munka2
$ chown :franko munka2
$ ls -l munka2
-rw-------  1 franko  franko  0 Feb 22 13:35 munka2
```

Ezzel a `munka2` állomány tulajdonosa és csoportja egyaránt `franko` lesz (a *FreeBSD* esetén minden felhasználónak alapból a saját nevével létrejön egy csoport is), illetve csak a tulajdonosnak van olvasás és írás joga. Ha újra lekérdezzük a *ACL* jogokat, sok változás nem fog történni a fent végrehajtott változásokat leszámítva:

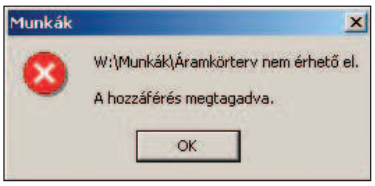
```
$ getfact munka2
#file:munka2
#owner:1001
#group:1003
user::rw-
group::---
other::---
```

A hozzáférést leíró táblázat szerint `noobi` és `joe` férhet hozzá az állományhoz olvasás joggal. Őket egyszerűen csak hozzá kell adni a listához, majd rögtön meg is nézzük, mi változott.

```
$ setfacl -m user:noobi:r-- munka2
$ setfacl -m user:joe:r-- munka2
$ getfacl munka2
#file:munka2
#owner:1001
#group:1003
user::rw-
user:joe:r--
user:noobi:r--
group::---
mask::r--
other::---
$ ls -l munka2
-rw-r-----+ 1 franko  franko  0 Feb 22 20:35 munka2
```

Az első különlegességet az `ls` parancs kimenetében találjuk, mégpedig egy + jel képében, amely azt jelzi, hogy a látott jogoknál sokkal több jogdefinícióra kell számítanunk, mint amennyit látunk. Ezek a jogok a `setfacl` parancs `-m` kapcsolójával kerültek oda, amely a *merge* rövidítése, így annyival több a szimpla hozzáadásnál, hogy nem ismétli meg az azonos sorokat. A `getfacl` listájában találkozunk is ezekkel a sorokkal, pont úgy, ahogy beírtuk őket. Az egyetlen ismeretlen sor a `mask::r--` lesz, amely annyit tesz, hogy a meghatározza a legmagasabb adható jogokat – leszámítva természetesen a tulajdonos saját *ACL* bejegyzését és az egyéb kategória jogait. Ez a maszk arra jó, hogy automatikusan mutatja a legmagasabb jogokat az *ACL* jogok közül, illetve azonos azzal amit az `ls -l` kiír a csoportra vonatkozó jogoknál. Megadhatunk csoportokra is jogokat, viszont törekedjünk arra, hogy tisztán csak felhasználókat adjunk hozzá a listához, bár sok olyan helyzettel találkozhatunk, mikor csoportokat egyszerűbb kezelni. Kipróbálhatjuk, hogy működik-e a megoldásunk, egyszerűen a megnevezett felhasználókkal kell próbálnunk olvasni, írni vagy futtatni a megadott állományt (csodálkoznék, ha nem működne helyesen). A `munka1` állomány jogait is gyorsan beállíthatjuk:

```
$ chmod 600 munka1
$ chown tomy:tomy munka1
$ setfacl -m user:tomy:rwx munka1
$ setfacl -m user:noobi:rwx munka1
$ setfacl -m user:franko:r-- munka1
$ getfacl munka1
#file:munka1
#owner:10636
#group:1008
user::rw-
user:franko:r--
user:noobi:rwx
user:tomy:rwx
group::---
mask::rwx
other::---
```



2. ábra Hozzáférés megtagadva...

```
$ setfacl -d -m user:tomy:rwx
➥Áramkörterv
setfacl: acl_calc_mask() failed:
➥Invalid argument
setfacl: failed to set ACL mask on
➥Áramkörterv/
```

Az ACL jogok másolása

Az egész játszadozás a jogokkal nem ér semmit, ha ezt minden egyes új állományra alkalmaznunk kell. Egyik megoldásunk erre az *ACL* „átmásolása" egy másik állományra:

```
$ getfacl munka1 | setfacl -M - munka2
```

Ezzel a módszerrel a `munka1` állomány *ACL* listája hozzámásolódik a `munka2` állomány *ACL* listájához. Ha ez utóbbi „üres" volt eredetileg, akkor a két állomány listája azonos. Ez látszólag félmegoldás, mivel az új állományokra egyesével meg kell oldani ezt a másolást. Megtehetjük, hogy egyik állomány jogait levesszük egy másik állomány listájáról, ekkor a

```
$ getfacl munka1 | setfacl -X - munka2
```

parancsot kell használnunk.

Az ACL jogok örökítése

Rövid és egyszerű munkákat leszámítva a valóságos projektek több állományt, esetleg egy egész könyvtárstruktúrát jelentenek. Ezeket az állományokat érdemes tehát egy közös könyvtárba tenni, és a könyvtárakkal is eljátszani a jogok kiosztásakor. Maradjunk meg a cikk eleje táján vázolt táblázatnál, csak az eddig használt állományok helyett egy könyvtárstruktúrát építünk ki, amely egy ***Munkák*** főkönyvtárból, benne egy ***Áramkörterv*** és egy ***Program*** alkönyvtárból. Érdemes a projekt könyvtárát arra a felhasználóra átállítani, aki a gazdája annak, így az ***Áramkörterv*** tulajdonosa `tomy` lesz, a ***Program*** könyvtáré pedig `franko`.
A `setfacl -d` kapcsolója felel az alapértelmezett *ACL* jogok kezelésért, amely öröklődik a könyvtárstruktúrában. Ha ezt a kapcsolót is megadjuk, akkor a hozzáfűzött lista lesz az összes – e könyvtár alatt – létrehozott állomány alapértelmezett *ACL* joga. Az alapértelmezett listát a `getfacl -d` kapcsolójával tudjuk lekérdezni:

```
$ getfacl -d Áramkörterv
#file:Áramkörterv
#owner:10636
#group:1008
```

A listánk láthatólag teljesen üres, ezért hozzá kell adnunk néhány jogot. Ha ezt a tevékenységet nem a ***UNIX*** jogokkal kezdjük, akkor az alábbi hibaüzenettel nézünk farkasszemet:

A megoldás egyszerűen annyi, hogy először és lehetőleg egyszerre hozzá kell rendelnünk azokat a jogokat, amelyek az állomány tulajdonosára és a csoportjára vonatkoznak:

```
$ setfacl -d -m u::rwx,g::---,o::---
➥Áramkörterv
```

Ekkor már hozzá tudjuk rendelni egyenként a felhasználók jogait a könyvtárhoz (ne felejtsük el, hogy a `franko` felhasználónak adjunk x jogot is, különben nem fog tudni belépni a könyvtárakba):

```
$ setfacl -d -m user:tomy:rwx Áramkörterv/
$ setfacl -d -m user:noobi:rwx Áramkörterv/
$ setfacl -d -m user:franko:r-x Áramkörterv/
$ setfacl -m user:tomy:rwx Áramkörterv/
$ setfacl -m user:noobi:rwx Áramkörterv/
$ setfacl -m user:franko:r-x Áramkörterv/
```

Az elkészült struktúrában már csak néhány próbát kell elvégezni, hogy láthassuk működés közben:

```
franko Munkák/Áramkörterv $ touch állomány
touch: állomány: Permission denied
franko Munkák/Áramkörterv $ mkdir könyvtár
mkdir: könyvtár: Permission denied
tomy Munkák/Áramkörterv $ touch állomány
noobi Munkák/Áramkörterv $ touch állomány2
tomy Munkák/Áramkörterv $ mkdir könyvtár
noobi Munkák/Áramkörterv $ mkdir könyvtár2
tomy Munkák/Áramkörterv $ touch állomány2
touch: állomány2: Permission denied
```

Hoppá... minden a tervek szerint működne, viszont egy apróságról megfeledkeztünk. Új állomány vagy könyvtár létrehozásakor az operációs rendszer néhány bitjét maszkolja a jogoknak. Alapértelmezésben ez `0022`, amely szerint a csoport és az egyéb felhasználók nem lesznek képesek írni a létrehozott állományokba, s ez az *ACL* listák nélkül tökéletesen megfelelő megoldás. Azonban ez hatással van az *ACL* maszkra is, így az újonnan létrehozott állományok `mask` sora is változni fog:

```
$ getfacl állomány
#file:állomány
#owner:10636
#group:1008
user::rw-
user:franko:r-x          # effective: r--
user:noobi:rwx           # effective: r--
user:tomy:rwx            # effective: r--
group::---
mask::r--
other::---
```

A megoldás egyszerű: vagy ideiglenesen megváltoztatjuk az `umask` parancs segítségével ezt a változót, vagy beépítjük a */etc/login.conf* állományba.
Nézzük meg a változást:

```
tomy Munkák/Áramkörterv $ umask 0002
tomy Munkák/Áramkörterv $ touch állomány3
noobi Munkák/Áramkörterv $ umask 0002
noobi Munkák/Áramkörterv $ touch állomány3
franko Munkák/Áramkörterv $ umask 0002
franko Munkák/Áramkörterv $ touch állomány3
touch: állomány3: Permission denied
```

Vissza vannak még a könyvtárak ellenőrzése, de ez már nem okoz gondot:

```
tomy Munkák/Áramkörterv $ mkdir könyvtár
noobi Munkák/Áramkörterv $ mkdir könyvtár2
franko Munkák/Áramkörterv $ mkdir könyvtár3
mkdir: könyvtár3: Permission denied
```

A könyvtárak létrehozására is működik az alapértelmezett listánk, próbáljuk ki a könyvtárváltást is:

```
tomy Munkák/Áramkörterv $ cd könyvtár2/
noobi Munkák/Áramkörterv $ cd könyvtár
franko Munkák/Áramkörterv $ cd könyvtár
```

Tehát mindenki megkapta a futtatás jogok a létrehozott könyvtárakra. A próbát oldjuk meg úgy, hogy `tomy` és `noobi` felhasználó egymás könyvtárába lépjen bele, ezzel kiderítjük, hogy képesek-e új állományt létrehozni, hogy a másik felhasználó a könyvtár tulajdonosa:

```
franko Munkák/Áramkörterv/könyvtár $ touch állomány
touch: állomány: Permission denied
tomy Munkák/Áramkörterv/könyvtár2 $ touch állomány
noobi Munkák/Áramkörterv/könyvtár $ touch állomány2
```

S végezetül az illetéktelen felhasználók jogait is érdemes megtekinteni:

```
auth.gabor Munkák $ cd Áramkörterv/
-bash: cd: Áramkörterv/: Permission denied
```

Kapcsolat a Samba programmal

Mindennek a sok szövegnek nincs értelme, ha nem tudnánk ezek után fájlszerverként használni rendszerünket. A *Samba* program képes az *ACL* támogatást kihasználni, csak úgy kell fordítanunk (vagy olyan lefordított binárist telepítenünk), hogy ezt tudatosítjuk is benne. Ezen túl más dolgunk nincs is, mint kipróbálni a megosztáson az említett jogokat.
Ha például egy *Windows* ügyfélen felcsatoljuk a megosztást (a példában `W:` meghajtóként), és lekérdezzük az *Áramkörterv* könyvtár jogait, akkor egyszerűen ugyan azokat a jogokat kell látnunk, mint amit kínkeserves munkával begépeltünk a parancssorba. Egy kicsit látványosabb és könnyebb is egy ilyen megosztáson át piszkálni a hozzáférési jogokat, a parancssori mókát hagyjuk a programocskákra és az esetleges apróbb módosításokra.
Az *ACL* működését ki is próbálhatjuk – szintén *Windows* ügyfél segítségével. A megosztást olyan felhasználó nevével csatoljuk fel meghajtóként, akinek nincs joga olvasni sem az *Áramkörterv* könyvtár tartalmát, majd próbáljunk belépni a megnevezett könyvtárba. Elvileg *„A hozzáférés megtagadva”* üzenetet kell látnunk a képernyőn, demonstrálva ezzel a jól működő rendszerünket.

Kiterjesztett attribútumok

Sokszor jönne jól, ha egy állományhoz megjegyzéseket tudnánk fűzni, például olyan képet hordozó állományhoz, amely formátum nem támogatja a megjegyzésmezőt. A kiterjesztett attribútumok pont erre a célra szolgálnak: hozzá tudunk fűzni kiegészítő adatokat az állományainkhoz.
Az *ACL* kezeléséhez hasonlóan itt a `getextattr`, a `setextattr`, az `lsextattr` és az `rmextattr` parancsok szolgálnak e tulajdonság kezelésére. Egy állományhoz több attribútum is rendelhető, ugyanis meg kell neveznünk az információt, amit tárolni szeretnénk. Ezen túlmenően két névtér közül kell kiválasztanunk a megfelelőt: `user` vagy `system`. Például a lementett nyers képhez hozzáfűztem a felhasználói névterületen egy `tartalom` nevű változóba a kép magyarázó szövegét:

```
$ setextattr user tartalom “Samba – Egy könyvtár
➥jogainak beállítása” Samba01.bmp
$ setextattr user idopont “`date`” Samba01.bmp
```

Meg tudjuk nézni, hogy milyen attribútumok vannak egy állományhoz rendelve:

```
$ lsextattr user Samba01.bmp
Samba01.bmp     tartalom        idopont
```

Illetve le tudjuk kérdezni a megnevezett attribútumot is:

```
$ getextattr user tartalom Samba01.bmp
Samba01.bmp     Samba ACL kezelése
```

Ha már nem szükséges, akkor törölhetjük is:

```
$ rmextattr user tartalom Samba01.bmp
```

Ezt az információs területet sok érdekes dologra felhasználhatjuk, bár a legtöbb formátum esetén maguk a programok oldják meg az ilyen információk kezelését az állományon belül.

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

KAPCSOLÓDÓ CÍMEK

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

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

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

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

Az eFltk bemutatása (1. rész)

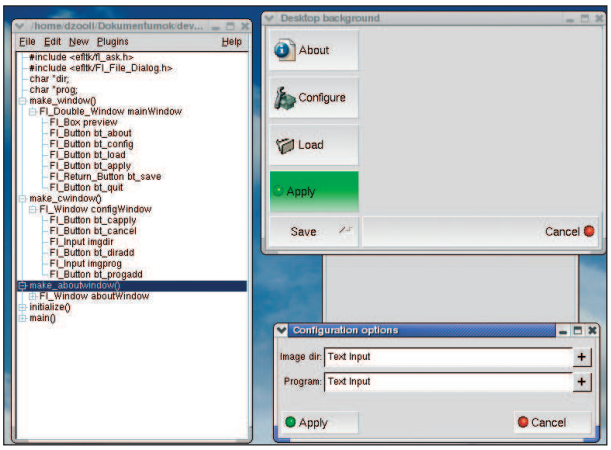
Egy nagyon jól használható grafikus eszközkészletet (widget-set) szeretnék bemutatni. A neve eFltk, ami egyben utal az elődjére is.

Első nekifutásra talán egy kis történelmi háttérrel kell kezdeni az ismerkedést és a grafikus felületek általános működésének tanulmányozásával. Általánosságban elmondható, hogy egy grafikus felület fő működése a megjelenítés után az események feldolgozásából áll. A mai grafikus felületek mind esemény-vezérelt módon működnek és ez igaz a ***Win32***, a ***Mac OS***, az ***Xwindow*** megjelenítésére egyaránt. A program fő tevékenysége az események figyelése és továbbítása a grafikus felület egyes elemei felé.

Az ***eFltk*** sem működik másképpen, azonban hosszú fejlődési időszak után tisztultak le a körvonalak és alakult ki az az eszközkészletet, amiről most szó lesz. A készlet elődjét ***Bill Spitzak*** kezdte el fejleszteni, aki korábban a ***NeXT*** grafikus felületének fejlesztésével foglalkozott 1987-ben. A korai változatot *„views”*-nek nevezték el, de sajnos zárt forráskódja miatt ma nem sokat tudhatunk róla. A következő állomás a ***Forms*** eszközkészletet megismerése és jó tulajdonságainak átvétele volt, majd kiegészítések váltak szükségessé az ***OpenGL*** és ***GLX*** kiegészítések használatához. Ebben az időszakban született meg valójában az ***Fltk*** első változata, ami alapját képezi jelenlegi témánknak.

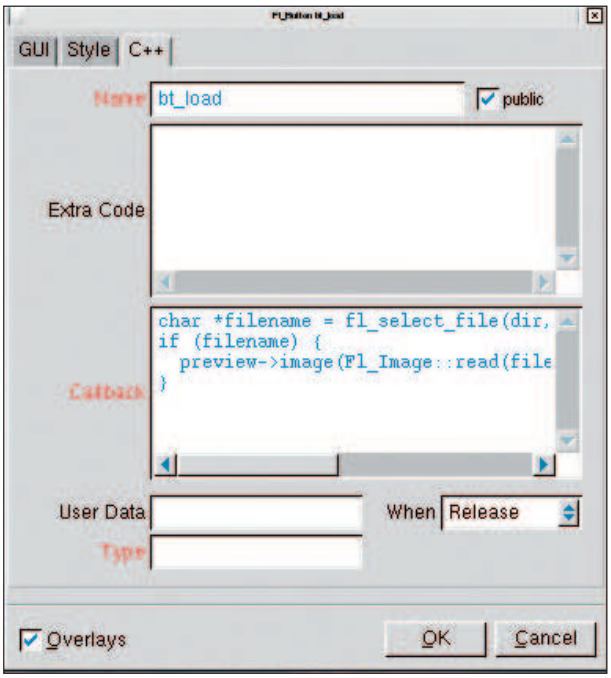
Magát az ***eFltk*** készletet jelenleg hárman fejlesztik egy nagyobb projekt részeként. A munka célja egy általánosan használható grafikus munkakörnyezet kialakítása, amely az ***eFltk***-ra épül és ebből következően gyors működésű és kis helyigényű alkalmazások összessége. A készülő munkakörnyezet tartalmaz, munkaasztal kezelést, ikonok megjelenítése lehetséges, része egy ablakkezelő, egy beállító alkalmazás, egy naptár, óra és egyéb segédprogramok. A környezet neve ***Equinox Desktop Environment*** vagy rövidebben ***ede***. A linkek között megtalálható az eredeti változat és a bővített változat is, amely már tartalmaz a tálcára elhelyezhető ikonokat is.

Ennyi bevezető után azonban tekintsük át az ***eFltk*** legfontosabb tulajdonságait. Elsősorban fontosnak tarom megemlíteni, hogy a készlet minden gond nélkül használható ***Win32***, ***Unix***, ***Linux*** és ***Mac OS X*** környezetben a megfelelő (***MSVC++***, ***GNU C***, ***MinGW***) fordítóprogramok képesek lefordítani készletet és a vele készült alkalmazásokat is. Tehát egy több-platformos grafikus eszközkészletről van szó, ami ***C++*** nyelven készült. Objektum-orientált működéséből adódóan azt gondolhatnánk, hogy a sebessége nem túl látványos, én azonban az elkészült alkalmazásokat és a példákat még így is gyorsabbnak találtam a ***GNOME*** vagy a ***GTK+*** alkalmazásoknál. Ezt talán annak köszönheti, hogy az eseménykezelő ciklusa egy egyszerűsített sémát követ. Az eseményeket mindaddig adja tovább a vezérlőknek, amíg azt valamelyik le nem kezeli vagy amíg van még vezérlő a grafikus elemek között. Nincs szükség annak meghatározására, mely eseményeket melyik vezérlők fogadhatják, hiszen mindegyik megkapja és vagy kezeli vagy visszautasítja. Sok-sok feltételvizsgálatot megtakaríthatunk ilyen módon, főleg ha egy bonyolultabb alkalmazásról van szó.

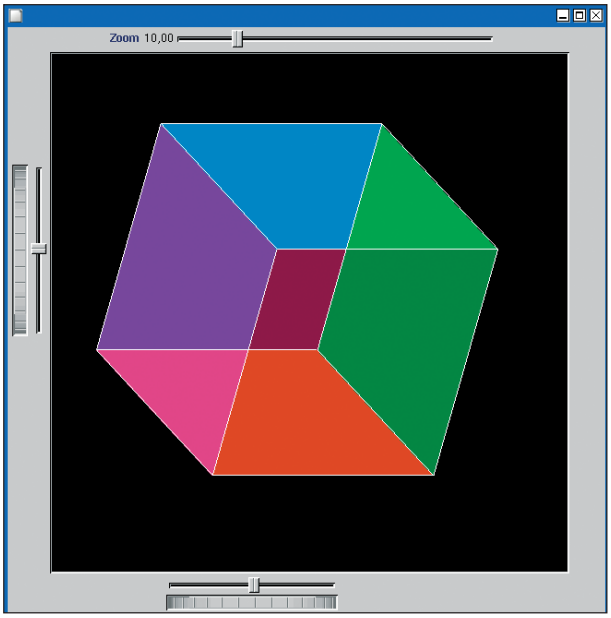


1. ábra Az eFluid felületkészítő

Az ***eFltk***-ban továbbá használhatunk ***OpenGL*** és ***GLX*** segítségével történő megjelenítést, nagyon sokféle vezérlő áll rendelkezésünkre és a fejlesztést nagyban megkönnyíti a grafikus fejlesztőkörnyezet az ***efluid***. Erről az alkalmazásról láthatunk egy képet az *1. ábrán*. Az alkalmazás nagyban felgyorsítja a fejlesztés menetét, hiszen gyorsan megtervezhetjük az alkalmazásunk grafikus felületét, majd a külső rutinok elkészítésével és használatával függetleníthetjük a grafikus megjelenítést a program többi részétől. Természetesen egyszerűbb alkalmazások esetében erre nincsen szükség és ilyen programokat akár szövegszerkesztő használata nélkül is készíthetünk az ***efluid***-



2. ábra Kódszerkesztő az eFluid-ban



3. ábra OpenGL alkalmazás eFltk-ban

ban. A program ugyanis lehetővé teszi, hogy az események kezelésére szolgáló kódrészleteket is a tervező felületen írjuk be a megfelelő helyre. A 2. *ábrán* látható egy egyszerű alkalmazás éppen a kódrészlet készítése közben.

Néhány mondat erejéig tekintsük át, hogy milyen kiegészítések találhatók az elődhöz képest az ***eFltk***-ban. Mindamellett, hogy az általános vezérlőelemeket megtaláljuk (gombok, listák, választógombok, görgetősávok, menü, felbukkanó menü és további vezérlőelemek), fontos megemlítenem, hogy a készlet alkalmas adatbázis kapcsolatok létrehozására is. Lehetőségünk van ***MySQL*** és ***ODBC*** adatbázisokhoz kapcsolódni, majd az egyes vezérlőelemek értékét a lekérdezések eredményéből meghatározni. Nagyon egyszerűen bővíthetjük további adatbázis kapcsolatokkal a meglévő készletet és találhatunk az adatbázis rekordjainak szerkesztésére alkalmas párbeszédablakot is.

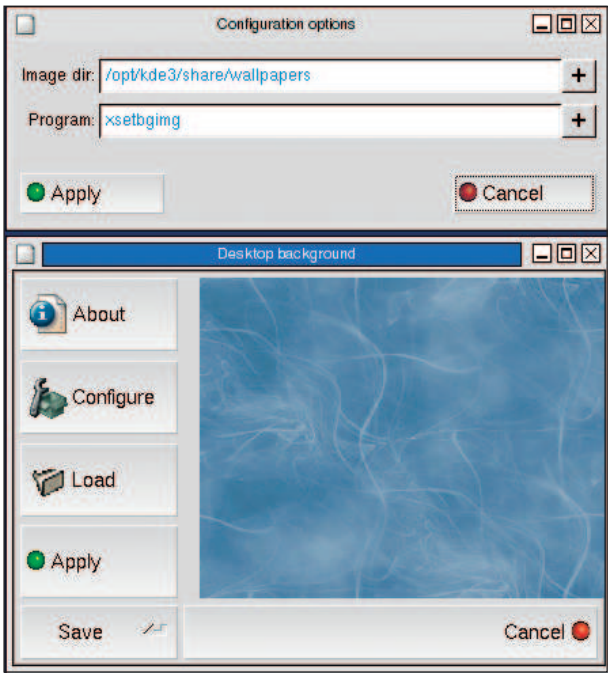
Egy jól használható eszközkészletből nem maradhat ki a hálózati kapcsolatok kezelésére szolgáló osztály sem, melyekből az ***eFltk***-ban többet is megtalálhatunk. Lehetőségünk van ***IMAP***, ***FTP*** és általános hálózati kapcsolat kezelésére a meglévő osztályok segítségével. Könnyedén elkészíthetjük az alaposztályból a számunkra megfelelő hálózati protokoll kezelésére alkalmas leszármazottakat is.

A mai grafikus felületekből nem maradhat ki a képek kezelése sem. Találhatunk beépített képeket, melyeket például alkalmazhatunk gombok vagy menüelemek látványosabbá tételére de a készlet alkalmas ***PNG***, ***BMP***, ***GIF***, ***JPEG*** és ***XPM*** képek betöltésére és megjelenítésére. A képek adatait közvetlenül is elérhetjük, ami lehetővé teszi, hogy képfeldolgozó alkalmazásokat készítsünk. Amennyiben nincs szükségünk különleges képkezelő műveletekre, a képek kezelését végző rutinokat ki is hagyhatjuk a kód szerkesztéséből így kisebb programokat készíthetünk. A későbbiekben vizsgálandó, *4. ábrán* látható program mindössze 77 kilobyte, amelyből a képek 43 kilobyte-ot foglalnak el a programban.

Említésre méltó, hogy a fejlesztők gondoltak a konfigurációs állományok kezelésére is, találhatunk olyan osztályokat, melyekkel a megszokott ***INI*** állományok szerkezetéhez hasonló beállításokkal dolgozhatunk. Az osztály egyszerűsíti a beállítások tárolását és betöltését, nagyban megkönnyítve ezzel mindennapi programozási feladataink ellátását.

Szintén az ***eFltk*** jól használhatóságát bizonyítja az is, hogy az előre elkészített osztályok között találunk naptárat, gyors képkezelésre alkalmas memóriában módosítható (offscreen) bitképeket, állomány választó párbeszédablakot, szövegszerkesztőt és a színek kiválasztására alkalmas párbeszédablakot is.

Egyetlen hátránya az elemkészletnek az, hogy a dokumentáció nem teljes. Az ismert, változatlan részek dokumentumai megegyeznek az eredeti ***Fltk*** dokumentációval, így ezeket nem találjuk meg a html formátumú dokumentumok között, ami újdonság, arról pedig nagyon gyakran csak az osztályfüggvények és változók nevét tudhatjuk meg. Ez bizonyos szinten érthető, hiszen nem mindenkinek egyértelmű és megszokott, hogy fejlesztés közben dokumentálja is a készülő kódot. Ez időigényes feladat, ugyanakkor nagyban könnyíti az érthetőséget és az áttekinthetőséget. Személyes tanácsom, hogy fejlesztés során alkalmazzunk valamilyen önműködő dokumentáció készítő programot, mint például a ***Doxygen***. A program fejlesztése közben így a kóddal együtt készülhet a dokumentáció és egyetlen parancs végrehajtásával elkészíthetjük a ***HTML***, ***RTF***, ***LaTeX*** formátumban elérhető dokumentációt is. A ***Doxygen*** segítségével néhány szabály elsajátítása árán akár magyar nyelvű, képekkel és hivatkozásokkal gazdagított dokumentációt is készíthetünk.



4. ábra Egyszerű eFltk alkalmazás

A program ugyanakkor alkalmas az osztályhierarchia megjelenítésére is, melynek segítségével könnyen érthetővé tehetjük a forráskódokat és az általunk készített új osztályokat is.
Az ismerkedést folytassuk most a klasszikus *„Hello World"* programmal. A grafikus felület megtervezéséhez használjuk az *efluid* programot, és készítsünk egy ablakot, melyben egy gomb felirata legyen *„Hello World"*. A grafikus tervezőprogram képes előállítani a *C++* nyelvű forráskódot is, amit az 1-es listán láthatunk.
A programban találhatunk három függvényt, melyek közül a `make_window` építi fel a grafikus felületet. Amint láthatjuk a felület felépítése nem bonyolult, azonban mégis azt javaslom, hogy a generált kódhoz csak akkor nyúljunk szövegszerkesztővel, amikor már egészen biztosak vagyunk abban, hogy a felület nem fog változni. Nagyobb alkalmazások esetében az események kezelésére szolgáló függvényeket célszerű külön forráskód állományban tárolni, és a tényleges eseménykezelőkből ezeket a külső programrészeket meghívni. Így elérhetjük, hogy a felhasználói felület független legyen a működéshez szükséges kódrészletektől. Tekintsük át most az első listát.

```
#include "hello.h"
Fl_Window* window;

static void cb_Hello(Fl_Button*, void*) {
 window->hide();
}

Fl_Window* make_window() {
 Fl_Window* w;
 {Fl_Window* o = window
 = new Fl_Window(280, 80, "Hello eFltk");
 w = o;
 o->shortcut(0xff1b);
 {Fl_Button* o
 = new Fl_Button(25, 15, 230, 45, "Hello World");
 o->label_font(fl_fonts+1);
 o->label_size(24);
 o->callback((Fl_Callback*)cb_Hello);
 }
 o->end();
 }
 return  w;
}

int main (int argc, char **argv) {

 make_window();
 window->show();
 return  Fl::run();
}
```

A listán jól látható, hogy a program fő függvénye mindössze három sorból áll. Az első sorban elkészítjük az ablak grafikus felületét, majd megjeleníti az ablakot. Ez után kezdődik a program eseménykezelő ciklusa, ami addig fut, míg valamilyen ablak látható a képernyőn. Az ablak eltüntetéséről a gomb eseménykezelő függvénye gondoskodik mégpedig oly módon, hogy a fő ablakot elrejti. Ennek hatására a program befejezi futását.
A program lefordítására létezik egy egyszerű módszer: a parancssorban adjuk ki az

```
efltk-config –compile hello.cpp
```

parancsot. Így statikusan összekapcsolhatjuk a programot a szükséges rutinokkal és bármilyen ***Linux*** környezetben működésre bírhatjuk.
A következő részben folytatjuk az ismerkedést az elemkészlettel, addig is mindenkinek kellemes munkát és tanulást kívánok.

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.

KAPCSOLÓDÓ CÍMEK

Az Ede (Equinox Desktop Environment) bővített változata:
➲ http://dzooli.uw.hu/ede-1.0.2-dzooli.tar.bz2

Az eredeti fltk:
➲ http://www.fltk.org

Az eFltk honlapja:
➲ http://ede.sourceforge.net

Az SMTP AUTH beállítása

IP cím ellenőrzés helyett hitelesítsük Postfix kiszolgálónk felhasználóit név és jelszó alapján – avagy ami nem állandó, az fix, hogy megváltozik.

Egyszer volt, hol nem volt, volt egyszer egy rendszergazda. A szemfüles olvasó bizonyára felfedezte azt, hogy az előző cikkem is ugyanezen felütésszerű mondattal kezdődött. Az egybeesés nem véletlen. Hősünk ebben a mesében sem lesz más, mint a mérnöki tudományok teljes fegyvertárát zsonglőrként alkalmazó rendszergazda, aki alig várja, hogy a felhasználók igényeinek kielégítése után következhessen a megérdemelt pihenés.

Eme jubileuminak mondható második epizód sem különbözik az elsőtől. A rendszergazda vígan tengeti életét, mígnem a felhasználók világvége hangulatot idéző sikoltásai nyomán szembekerül a problémával. Első benyomásra a problémának hét feje van és lángok csapnak fel szörnyűséges torkából, de közelebbről megvizsgálva kiderül, hogy csak egy pici gyíkról volt szó. A szükséges intézkedések nyomán hősünk végleges megoldást talál, és láthatjuk a könnyű nyáresti szellő borzolta hajkorona sziluettjét a lemenő nap utolsó sugaraiban.

Ne szaladjunk azonban előre! Rendszergazdánk munkahelyén a levelező kiszolgálót helyi hálózaton keresztül érik el a munkaállomások. Természetesen ugyanennek a számítógépnek van egy lába az Internet felé is, valamint egy *DNS*-ben jegyzett *MX* rekordja. Ez egy igen kellemes kialakítás, hiszen a kötelező biztonság mellett egyszerűen beállítható és karbantartható. Ne felejtsük el, hogy ha lehet választani, hősünk mindig az egyszerűbb megoldás mellett dönt.

A levélküldő kiszolgáló egy előfordított *Debian* csomagból telepített *Postfix*. Miután a cég összes munkaállomása a helyi hálózaton van, kézenfekvő beállításnak tűnt a ***levéltovábbítás (smtp relay)*** engedélyezése a teljes belső hálózatra. Miután történetünk rendszergazdája felvette az alábbi sort a */etc/postfix/main.cf* fő konfigurációs állományba, az e-mail küldés már működött is. Csupán némi finomhangolás maradt hátra, melyet itt most nem részleteznék.

```
mynetworks = 127.0.0.0/8, 192.168.0.0/24
```

Telt, múlt az idő, és a cég vezetősége egy új iroda létrehozását tűzte ki célul. Nem tervezték azonban külön *SMTP* kiszolgáló felállítását a létesítményben. Feltételezték, hogy egy közönséges *ADSL* kapcsolat előfizetésével a levelezés kérdésére megadták a választ. Sebtében kiadták a parancsot hősünknek, nevezetesen „legyen e-mail”. Bár a rendszergazdák isteni származása megkérdőjelezhetetlen tény, és így bizonyításra nem szorul, sajnos ezúttal nem élhetek a szent könyvekből ismert pátoszos fordulattal, miszerint „és lőn”.

A gondot az jelentette, hogy az *ADSL* előfizetés dinamikusan változó *IP* címre szólt. Mikor rendszergazdánk felvetette ezt a problémát a cég vezetőinek, arra kellett rádöbbennie, hogy minden jó szándék mellett a pánikhangulatnál többet nem tud elérni náluk. Így számba vette a szóba jöhető ötleteket és elkezdte latolgatni a várható előnyöket és hátrányokat. Négy gondolatfoszlány suhant át az agyán.

Az első, és egyben magától értetődő ötlet a fentebb említett `mynetworks` sor bővítése. Mivel a *Postfix* alapvetően *IP* cím alapján azonosít, elképzelhető, hogy ha egy kellően tág tartományban engedélyezett a levéltovábbítás, akkor ezzel hősünk átvágta a gordiuszi csomót. A kellően tág tartomány jelen esetben azt jelentené, hogy fel kell venni azt a teljes IP cím tartományt, amelyből a távoli iroda ***internet-szolgáltatója (ISP, Internet Service Provider)*** *IP* címet oszthat.

Égbekiáltó gaztett lenne a fentebb vázolt kósza gondolat kivitelezése. Ezáltal az összes, ugyanahhoz a szolgáltatóhoz tartozó vadidegen előfizetőnek engedélyezve volna a levélküldés. Ez nem pusztán biztonsági rés, de könnyen meg is utáltathatja vele magát a meggondolatlan adminisztrátor. A levélszemetet ontó kiszolgálót nem igazán szeretik az Internet harcedzett használói. Mi több, egy ilyen merénylet elkövetése után a kiszolgáló teljesen jogosan bekerülhet a ***Nyitott Átjárók Adatbázisába (Open Relay Database, http://www.ordb.org/)***, ami által számos e-mail kiszolgáló eleve el fogja utasítani az összes, tőle érkező levelet.

Második lehetőségként felmerülhet a dinamikusan osztott *IP* kézzel történő felvétele a fenti sorba. Persze ami kézi szerkesztéssel kivitelezhető, az megoldható szkripttel is. Még azonban így is rá kell bírni a távoli iroda átjáróját arra, hogy valahogyan tudassa az új címet, ezután újra kell tölteni a *Postfix* konfigurációt. Ez az elképzelhető legkörülményesebb, és túl sok helyen támadható. Néha be kell látni, hogy az egyszerűbb lehet bonyolultabb is.

Az első két gyenge próbálkozásra fátylat borítva hősünknek eszébe ötlött, hogy látott már olyan levelező programot, ami támogatta a ***POP-before-SMTP*** nevű eljárást. Ez a körültáncolt problémára egy olyan csellel nyújt megoldást, hogy a levelező program küldés előtt feljelentkezik a ***POP3*** kiszolgálóra. Miután ott hitelesítette magát, és az ***SMTP*** kiszolgáló erről tudomást szerzett, a felhasználó szabadon küldheti leveleit. Ezt azonban nem minden levelező ügyfél támogatja.

A múlt megnyugtató homályába veszett gondolatkísérletek után rendszergazdánk rátalált a legelfogadhatóbb megoldásra. Az eljárás neve ***SMTP AUTH***, és arról szól, hogy minden ügyfél az IP cím helyett egy név és jelszó páros segítségével nyer hitelesítést. A levelező programokban csak egy ezt az információt kell megadni, a kiszolgáló helyes beállítása mellett innentől zavartalanul folyhat az e-mail küldés.

Ennek a módszernek megvan az a határtalan szépsége, hogy hosszú távú megoldást biztosít. Egy újabb telephely létrehozása után minden további beállítás nélkül használható a levelező kiszolgáló. Mindössze a megfelelő felhasználókat kell létrehozni. Természetesen ezeknek nem kell rendszerfelhasználóknak lenniük, de ez egy másik mese. Rendszergazdánk megtalálta tehát a megoldást. Ezek után lássuk, mi hogyan kivitelezhetjük az ötletet.

A ***SASL*** a ***Simple Authentication and Security Layer*** rövidítése, és a 2222-es számú ***RFC*** taglalja részletesen. Érdemes a ***Cyrus-SASL*** nevű megvalósítás használatában elmélyedni, ha másért nem, azért, mert temérdek dokumentáció áll rendelkezésre az Interneten a szoftver használatához. Számos hitelesítési forrást kezel, sajnos azonban ***SQL*** adatbázisból, vagy ***LDAP*** kiszolgálótól egyelőre nem tudja közvetlenül lekérdezni a szükséges információkat. Ilyen jellegű igény esetén a forráskód foltozása az egyetlen út.

Telepíteni kell tehát egy ***Cyrus-SASL*** nevű szoftvert, valamint biztosítani kell, hogy a ***Postfix*** képes legyen a használatára. ***Debian*** alatt ez egyszerűen a ***postfix-tls*** csomag kiválasztásával elérhető. Mivel ez a csomag függ a megfelelő ***SASL*** csomagoktól, és fel is ajánl számos olyat, ami hasznos lehet a későbbiekben, érdemes az összes szintű függőséget kielégíteni. Miután szélsebesen feltelepítettünk mindent, következhet a beállítás.

Az első és legfontosabb teendő a meglévő ***main.cf*** állomány biztonsági másolatának elkészítése. Miután meggyőződtünk róla, hogy minden, amit ezután teszünk, visszafordítható folyamatot jelent, ragadjunk klaviatúrát és egy sokat próbált szövegszerkesztőnk segítségével nyissuk meg a konfigurációs állományt. Az első, amit a ***Postfix*** tudtára kell hozni, az az hogy használnia kell a ***SASL***-t.

```
smtpd_sasl_auth_enable = yes
```

Ezzel a sorral mellesleg remekül lehet ellenőrizni a ***postfix-tls*** csomag jelenlétét, illetve forráskódból történő telepítés esetén a megfelelő fordítás előtti beállítások helyességét. Ha a ***Postfix*** hibaüzenetet dob a fenti sorra hivatkozva, biztosak lehetünk benne, hogy hiányzik valamilyen összetevő a rendszerből. Következzen ezután egy biztonsági beállítás.

```
smtpd_sasl_security_options = noanonymous
```

Ezáltal kizárható a névtelen bejelentkezés lehetősége, mivel az ***SMTP*** kiszolgáló fel sem fogja ajánlani az ügyfélnek a hitelesítés ezen módját. A hitelesítési mód a sima szöveges bejelentkezéstől kezdve a ***Kerberos***-ig sokféle lehet, beleértve az imént kizárt módot is. Egy bizonyos hitelesítési forrás használatakor, melyre később még visszatérünk, szükséges a következő.

```
smtpd_sasl_local_domain = $myhostname
```

Ez egyfajta tartománynevet állít be, de hangsúlyozom, hogy alább még szó esik erről. Fontos továbbá a régebbi levelező programok támogatása is.

```
broken_sasl_auth_clients = yes
```

A ***Microsoft Outlook Express 4***, illetve a ***Microsoft Exchange 5.0*** két olyan meglehetősen régi levelező, amellyel az ***SMTP AUTH*** kommunikáció a szigorú szabványok megtartása mellett nem működik. A fenti sor ezen a gondon segít, és egyes szabványok enyhébb figyelembevételével lehetővé teszi azoknak az ügyfeleknek is a hitelesítést, melyek már kifejezetten korosnak mondhatók. Végezetül határozzuk meg, hogy a ***SASL*** hitelesített felhasználók küldhetnek a kiszolgálón keresztül levelet.

```
smtpd_recipient_restrictions =
    permit_sasl_authenticated,
    permit_mynetworks,
    check_relay_domains
```

Látható, hogy a kiszolgáló a `mynetworks` sorban szereplő ***IP*** címekről továbbra is fogadja a levelet, egy másik lehetőség a hitelesítés. A fenti változtatások eszközölése után újra be kell olvastatni a ***Postfix*** beállítási állományait. Ezt az alábbi paranccsal érhetjük el.

```
# /etc/init.d/postfix reload
```

A hitelesítés már működik is! Csak a hitelesítés forrása nincs meghatározva, ez viszont már csak ujjgyakorlat. A szigorúan hitelesítéssel kapcsolatos beállítások a ***/etc/postfix/sasl/smtpd.conf*** nevű állományban találhatók. Könnyen elképzelhető, hogy sok rendszeren ez az állomány nem létezik, ekkor kézzel kell létrehozni. Az is lehet, hogy a ***/usr/lib/sasl*** könyvtárban kell keresni a nevezett fájlt. Ez a helyzet többek között akkor, ha forráskódból történik a telepítés.

Az állomány legfontosabb paramétere a `pwcheck_method`. Valójában nincs is olyan sok beállítási lehetőség, de erre nem is volna szükség. Ezzel az egy paraméterrel a hitelesítés forrása könnyűszerrel állítható. A következő sorral meghatározzuk, hogy `sasldb`-t használunk erre a célra.

```
pwcheck_method: sasldb
```

Ez egy igen tipikus felhasználás. Nincs szükség az azonosításhoz külön démonra, ugyanis egy Berkeley adatbázisból nyeri a felhasználónevet, illetve a jelszót.

Az adatbázis a */etc/sasldb* néven található. Ebben a név és jelszó mellett még egy ***tartománynév (realm)*** is szerepel. Ennek az az oka, hogy ha egy levelező kiszolgáló több tartományért is felel, fontos lehet megkülönböztetni az egyik tartományban szereplő felhasználót a másik tartománybeli azonos nevű felhasználótól. Fontos, hogy a sasldb használatakor a hitelesítés csak akkor sikeres, ha a *Postfix main.cf* állományának `smtpd_sasl_local_domain` paraméterében ugyanazt a tartomány szerepel, mint ami az adott felhasználónévhez tartozik ebben az adatbázisban.
Már csak egy dolog maradt hátra. Hozzunk létre egy felhasználót, aki jogosult használni az *SMTP* kiszolgálót, *IP* címétől függetlenül. `sasldb` használatakor ez a következő paranccsal tehető meg.

```
# saslpasswd -c -u mail.vallalat.hu geza
```

Ezzel létrehoztunk egy */etc/sasldb* nevű állományt, ha az még nem létezett, és szerepel benne az új bejegyzés. A felhasználó neve `geza`, a `mail.vallalat.hu` tartomány tagja (realm), jelszava pedig a `saslpasswd` által bekért szó.
A `sasldb` egy *Berkeley* adatbázis, ezért ne számítsunk arra, hogy közönséges szövegmegjelenítővel emészthető formában látjuk a tartalmát, viszont a `sasldblistusers` segítségével remekül lehet böngészni is.

```
# sasldblistusers
```

Felhívnám a figyelmet arra, hogy ha a *Postfix* gyökérkönyvtárat vált indulás után (*chroot*), ennyivel még nem fogja megtalálni */etc/sasldb* néven az adatbázist. Ez a helyzet az előfordított Debian csomag esetén is. Ez esetben az állományt másoljuk át a *Postfix* új gyökerébe.

```
# cp /etc/sasldb /var/spool/postfix/etc/
➥ sasldb
```

Elkészültünk. Természetesen a hitelesítési forrás megváltoztatásával elérhető, hogy ne legyen szükség külön felhasználói névre és jelszóra az *SMTP*-hez. Például *Cyrus IMAP* kiszolgáló esetén hitelesítési forrásként a `pwcheck`-et megadva az *IMAP*-es név és jelszó páros is minden további nélkül használható. További lehetőség a `shadow` érték, amivel a rendszeradatbázist foghatjuk munkára. Ez azonban komolyan ellenjavallt, amellett, hogy gyökérváltás után el sem érhető.

Sok sikert a beállításokhoz és biztonságos levelezést mindenkinek!

Fülöp Balázs (admin@guardware.com)
21 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.

MySQL 5 – rég várt fejlesztések

Megnézzük, hogy állnak a „legnépszerűbb nyílt forrású adatbázis-kezelő” programmal a fejlesztők.

Lassan 10 éve, hogy a *MySQL* töretlen pályája elkezdődött. A siker talán annak volt köszönhető (és ez így van a mai napig), hogy a pehelysúlyú kategóriát célozták meg a fejlesztők. Ennek a döntésnek a legfőbb előnye, hogy a *MySQL* egyszerű és gyors, könnyen megtanulható a használata, ideális a mindennapos igények kielégítésére. Hátrányként jelentkezik azonban, hogy a szolgáltatások köre szűkebb, mint a vetélytársaké. Ez ellen úgy próbáltak védekezni, hogy szép fokozatosan bővítették a lehetőséget, közben ügyelve rá, hogy ha nincs rájuk szükség, akkor épp olyan egyszerűen lehessen használni az adatbázis-kezelőt, mint annak előtte.
Az adatbázis-kezelés témakörében már jó ideje jelen lévő tárolt eljárások megjelenésével a *MySQL* készítői régi adósságukat törlesztették. Cikkünkben ezt, és egyéb, a témához kapcsolódó fejlesztéseket vizsgálunk meg közelebbről.

Helyzetkép

A cikk írásának pillanatában az 5.0.2-alpha fejlesztői változat érhető el „legfrissebb kiadás” címén a *MySQL* honlapján. Az egyes lehetőségek megvalósítása és beépítése a programba folyamatosan történik, minden kiadás tartalmaz egy-két újítást, s mellette rengeteg hibajavítást. A végleges változat tehát nem csak annyival fog többet tudni, hogy alaposan le lesz tesztelve, de jónéhány, jelenleg nem szereplő szolgáltatással is bővülni fog. Nekünk azonban jelenleg meg kell elégednünk a fenti változatszámú fejlesztői kiadással, de aggodalomra semmi ok, hisz ez is megfelelően körvonalazza számunkra, hogy egy esetleges következő alkalmazásunknál milyen lehetőségeket vehetünk igénybe, ha a *MySQL* adatbázis-kezelőt választjuk.

Nézzük az elsőt: nézetek

Az 5-ös változatban már lehetőségünk nyílik ***nézetek (view)*** készítésére. Egy ilyen nézet nem más, mint egy *SQL* lekérdezés által reprezentált adathalmaz, amelyet úgy érhetünk el, mintha az egy önálló tábla volna. Jó hír, hogy a nézetek mindjárt irhatók is, tehát lehetőségünk van a ***rekordok módosítására (update)***, illetve új sor ***beszúrására (insert)***, bár ez utóbbi bonyolultabb nézetek esetén nehézségekbe ütközhet, de ilyenre egyébként is csak a legritkább esetben vetemedünk.
A nézetek készítése viszonylag szabványosan történik, de van néhány megkötés. A nézet adathalmazát meghatározó lekérdezés nem tartalmazhat származtatott táblákat, nem hivatkozhat felhasználói változókra, nem hivatkozhat átmeneti táblákra, és ami az egyik legfontosabb: nem hozhatunk létre triggereket (lásd később) a nézet soraira.

Származtatott táblák, allekérdezések

A származtatott táblákat a 4.1-es változatban vezették be a készítők. Ez nem más, mint egy lekérdezés eredménye, amelyet felhasználunk egy másik, külső lekérdezésben mint táblát. Legfőképp abban különbözik a nézettől, hogy ez csak az adott lekérdezés idejéig létezik.
Példa:

```
select nev from (select azonosito,nev,cim from
➥szemelyek where azonosito>100) t2;
```

Ennek a lekérdezésnek persze a világon semmi értelme, de a származtatott tábla fogalma azért érzékelhető.
A fentihez hasonló dolog az ***allekérdezés (subquery)*** fogalma, amely a `WHERE` ágban elhelyezett lekérdezésre vonatkozik. Ez általában csak egyetlen mezővel tér vissza, s az alábbi formában szoktuk használni:

```
select nev from szemelyek where nev not in
➥(select nev from tiltott_szemelyek) tsz;
```

A példa egy negatív ***illesztést (join)*** hajt végre a két táblán, persze nem a leggyorsabb módon.

Tárolt rutinok

Egy tárolt eljárás olyan *SQL* nyelven írt parancsok összessége, amelyet az adatbázis-kiszolgálón rögzítünk, s utána a programozásban használatos módon meghívhatjuk. Ekkor elvégzi azokat az *SQL* utasításokat, amelyeket az eljárás törzsében meghatároztunk. Néhány bővítménynek köszönhetően tehát gyakorlatilag programozhatunk az adatbázis-kezelő berkein belül.
A megoldás egyik nagy előnye, hogy az adatbázissal kapcsolatos logika valóban az adatbázis szintjére kerülhet, nem teszi nehezen érthetővé a programkódot. A másik, talán ennél is fontosabb előny, hogy az itt elvégzett műveletekben szereplő rekordok nem hagyják el az adatbázis-kiszolgálót, hanem ott helyben történik meg a feldolgozásuk. Ennek következtében nem csak a hálózati terhelés csökken, de

Felhasználói változók

A *felhasználó változók (user variables)* olyan *MySQL* elemek, amelyekben a programozásban megszokott módon mindenféle értékeket tárolhatunk, s később hivatkozhatunk rájuk. Az ilyen változók jellegüket tekintve globálisak az adott kapcsolaton belül. Ez azt jelenti, hogy az adott ügyfélkapcsolaton belül bármelyik adatbázis használata közben elérhető, viszont senki más nem láthatja az általunk beállított változókat. Segítségükkel a különböző utasítások között rögzíthetünk állapotokat, teremthetünk kapcsolatot. Egy változó értékét a `SET @változónév = érték` paranccsal állíthatjuk be, mintha csak `UPDATE` művelettel volna dolgunk, az értékek kiolvasását pedig a `SELECT @változónév` paranccsal tehetjük meg, vagy simán a nevével hivatkozhatunk rá – például egy `WHERE` ágban.

mivel nincs késleltetés, maga a művelet is sokkal gyorsabb, mint a hagyományos esetben, arról már nem is beszélve, hogy az adatbázis-kiszolgáló ki tudja használni a saját maga által nyújtott gyorsítási lehetőségeket (például indexek).
A *MySQL* a tárolt eljárások kezelése során az *SQL:2003* szabványt követi, hasonlóan az *IBM* által fejlesztett *DB2* adatbázis-kezelőhöz.
Ennek megfelelően van *tárolt eljárásunk (stored procedure)* és van *tárolt függvényünk (stored function)*, amelyeket a leírás közös néven *tárolt rutinként (stored routine)* emleget. A kettő között a lényegi különbség az, hogy a függvénynek van valódi visszatérési értéke. Na de nézzük ezt meg részletesebben.

Az eljárások

Olyan visszatérési érték nélküli eljárásokról van szó, amelyek tetszőleges számú paramétert fogadva azoknak megfelelően végrehajtanak egy utasításhalmazt. Eddig nincs is benne semmi különös, a dolog ott kezd érdekessé válni, amikor megnézzük ezeket a paramétereket. Minden paraméter alapvetően háromféle lehet: `IN`, `OUT`, és `INOUT` (KI, BE és KI-BE) típusú, amelyek azt határozzák meg, hogy az adott paraméter bemeneti paraméter-e, vagy kimeneti, esetleg kétéltű. A bemeneti paraméterek a programozásban megszokott paraméterek, ezeket használjuk a függvényen belül, a kimeneti típusú paraméterek pedig visszatérési értéket képeznek. Egy ilyen kimeneti típusú paraméter csakis *felhasználói változó (user variable)* lehet, míg a bemeneti paraméter lehet konstans érték is (pl. egy szám).
A jelleg és a paraméter név megadása után következhet a típus megadása, ami jelen pillanatban a *MySQL* által használt típusok (oszlop típusok) lehetnek.
A tárolt eljárások a `CREATE PROCEDURE` parancs segítségével deklarálhatók, és ha egynél több utasítást tartalmaz, akkor `BEGIN` és `END` kulcsszavak közé kell ágyazni az eljárás törzsét. Elkészülte után a `Call <tárolt eljárás neve(paraméterek)>` módon hívhatók.
További hasznos érdekesség az eljárásoknál, hogy használhatjuk benne a jól megszokott `SELECT`-es lekérdezéseket, amelyek a hagyományos esetben visszaadódnak, mint az eljárás eredményei, tehát az eljárás futtatása ilyen esetben egyenértékű egy lekérdezés futtatásával. Fontos azonban megjegyezni, hogy ha több `SELECT` is lefut egy tárolt eljáráson belül, azok eredményei külön keresési eredményként adódnak vissza, tehát jó, ha a kliens tudja kezelni az összetett lekérdezési eredményeket.
Lássunk egy egyszerű példát az eljárások alkalmazására:

```
CREATE PROCEDURE negyzet(IN szam INT,OUT negyzete
➥ INT)
BEGIN
        SELECT szam*szam into negyzete;
END

Az eljárást az alábbi módon hívhatjuk:
mysql> CALL negyzet(4,@a);
Query OK, 0 rows affected (0.07 sec)
mysql> SELECT @a;
+------+
| @a   |
+------+
| 16   |
+------+
1 row in set (0.00 sec)
```

A példában a `SELECT INTO` szerkezetet alkalmaztuk, amely azt tudja, hogy a `SELECT` által visszatért értéket elhelyezi az általunk megadott változóba.

Függvények

A függvények rendelkeznek valódi visszatérési értékkel, s segítségükkel olyan számítási műveleteket végezhetünk, amelyek nem kapcsolódnak az adatbázishoz, illetve az ott tárolt adatokhoz, ugyanis egy függvényen belül nem használhatjuk a `SELECT`, `INSERT`, `UPDATE` és a hasonló műveleteket, tehát itt csak a kapott paraméterekkel dolgozhatunk, amelyek itt mindig bejövő értékeket takarnak.
Hasonlóan az eljárásokhoz a függvények a `CREATE FUNCTION` parancs segítségével deklarálhatók, és ha egynél több utasítást tartalmaz, akkor `BEGIN` és `END` kulcsszavak közé kell ágyazni az függvény törzsét.
Minden függvénynek van visszatérési-érték típusa, amelyet a deklaráció során adunk meg. Hasonlóan az eljárásokhoz, mind a függvény paraméterei, mind a visszatérési érték típusa *MySQL* által használt típus kell legyen.
Lássunk egy példát a függvények alkalmazására is:

```
CREATE FUNCTION funcdemo(szam int) RETURNS INT
BEGIN
RETURN szam*szam;
END
```

A függvényt az alábbi módon hívhatjuk:

```
mysql> select funcdemo(4);
+-------------+
| funcdemo(4) |
+-------------+
|          16 |
+-------------+
1 row in set (0.00 sec)
```

Megjegyzés: Mind az eljárások, mind a függvények esetében a törzs több BEGIN-END blokkból állhat, amelyek mindegyike felcímkézhető.

Vezérlési szerkezetek

A tárolt rutinok önmagukban még nem jelentenének nagy áttörést, azonban az elérhető vezérlési szerkezeteknek köszönhetően valóban programozhatunk *SQL* nyelven. Van IF, van CASE, REPEAT és WHILE ciklus, nincs FOR ciklus, van viszont egy érdekes LOOP nevű szerkezet, amely azt tudja, hogy a hurok kezdete és vége közötti részt ismételgeti, egészen addig, amíg a hurkon belül a megfelelő LEAVE utasítással a hurok elhagyására késztetjük a végrehajtást. Ezeken kívül létezik még egy ITERATE parancs, amivel a különböző ciklusokon belül kezdeményezhetjük a ciklus ismételt végrehajtását. (Az egyes szerkezetek szintaktikája megtekinthető a *MySQL* honlapján.)
Az itt felsorakoztatott lista teljes, és bár igen kevésnek tűnik, az a tapasztalatom, hogy a gyakorlatban elegendő, és az összes felmerülő probléma elvégezhető az elemek kombinálásával.

Változók, elemek

Csupán a vezérlési szerkezetekkel még nem sokra mennénk, a programíráshoz változók is kellenek, és a felhasználói változók nem feltétlenül alkalmasak erre a célra.
A már emlegetett BEGIN-END blokkon belül azonban lehetőségünk van különböző *MySQL* elemek és változók deklarálására. Nézzük meg közelebbről ezeket az elemeket:

- ***Feltételek (CONDITION)***: Segítségükkel hibakódokhoz rendelhetünk neveket, amely nevekre aztán később hivatkozhatunk, amikor le akarjuk kezelni a feltételek által reprezentált hibát
- ***Hibakezelők (HANDLER)***: Ezeket az elemeket ***kivételkezelésre (exception)*** használhatjuk, ahol a kivételt a *MySQL* kiszolgáló dobja egy hibakód formájában, mi pedig egy ilyen hibakezelő segítségével rögzíthetjük, hogy az adott hibakód felmerülése esetén milyen műveletet kell végrehajtani.
- ***Változók (VARIABLES)***: Ezek a hagyományos értelemben vett lokális változók, amelyek *MySQL* által használt típusúak lehetnek és rendelkezhetnek kezdőértékkel, de nem azonosak a felhasználói változókkal.

Mindhárom elemtípust a DECLARE kulcsszóval kell bevezetni, a pontos szintaktika a *MySQL* leírásban található.

Kurzorok

A kurzor nem más, mint egy lekérdezés eredményének olyan átmeneti tárolási lehetősége, ahonnan később akár rekordonként is visszahozhatjuk az abban tárolt adatokat. *MySQL*-ben minden kurzorhoz tartozik egy lekérdezés, amelyet a deklaráció során kell megadni. A kurzort a későbbiekben egy változó reprezentálja, amelyet megnyithatunk, értékeket olvashatunk ki belőle, majd lezárhatunk. Természetesen ezt is a DECLARE kulcsszóval vezethetjük be, és csak a tárolt rutin határait jelző BEGIN-END blokkon belül érvényes.
Van egy olyan szabály, miszerint nem mindegy, hogy milyen sorrendben deklaráljuk a fentebb emlegetett elemeket: hibakezelő után nem deklarálhatunk kurzort, és a változók deklarációjának mind a hibakezelők, mind a kurzorok deklarációját meg kell előzniük. Ha ebből deriválunk, az alábbi sorrendet kapjuk: változók, kurzorok, hibakezelők.
Kicsit térjünk most vissza a tárolt eljárásokhoz, s lássuk, hogy a fentieknek mi köze az egészhez. A kurzorok nélkülözhetetlenek az eljárásokban lekérdezett eredményhalmazok ciklikus bejárása érdekében, itt ugyanis nincs FOR SELECT (amely végigmegy egy lekérdezés eredményhalmazán), mint más alkalmazások esetében. A dolgot úgy kell áthidalni, hogy készítünk egy kurzort, amely tartalmazza a lekérdezés eredményeit, majd ezen kurzor minden elemén végighaladva járhatjuk be az eredményeket. A ciklus akkor ér véget, amikor elérjük a kurzor végét. Ezt – jobb híján – egy hibakezelő létrehozásával kell megoldani, amely bebillent egy jelzőbitet, amit a REPEAT-UNTIL ciklusban figyelünk. Még mielőtt megijednénk, hogy ez milyen bonyolult (bár más *SQL* megoldásokhoz képest határozottan az), lássunk egy példát a fentire:

```
create procedure curdemo()
BEGIN
DECLARE a int;
DECLARE b char(255);
DECLARE done INT DEFAULT 0;
DECLARE cur1 CURSOR FOR SELECT id,nev from
➥automarkak;
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET
➥done = 1;
OPEN cur1;
REPEAT
    FETCH cur1 INTO a, b;
    IF NOT done THEN
                SELECT a as id, b as nev;
    END IF;
  UNTIL done END REPEAT;
END
```

Ez a példa semmi hasznosat nem csinál, csupán kiír néhány autómárkát és nevet, sajnos mindet külön lekérdezés eredményeképp. A fentebb leírt módszert viszont jól szemlélteti. A done nevű változó jelzi, ha elfogytak az eredményhalmazból a rekordok, mivel a 02000-s SQLSTATE akkor következik be, ha az eredményhalmaz végére értünk.

Kis összegzés

Bár még nem értünk a cikk végére, hadd húzzak itt egy vonalat, és írjam le, ami a tesztelés során felhalmozódott bennem. Több helyen is le van írva a *MySQL* dokumentációjában, hogy a rendszer még fejlesztés alatt áll, de azt hiszem, hogy ha nem lenne leírva, erre akkor is hamar rájönnénk. Igen szegényes az elérhető lehetőségek köre. A legfőbb nehézség az, hogy nem tudunk úgynevezett leválogatást készíteni, amely valamilyen szempontok szerint kiszűrné egy lekérdezés eredményét, és csak a kritériumoknak megfelelő rekordokat adná vissza, ugyanis az eljárásnak nincs valódi visszatérési értéke, a SELECT által visszaadott értékek viszont soronként különböző lekérdezési eredményként adják vissza a kívánt rekordokat. Nem lehet típust definiálni, amivel áthidalhatnánk a visszatérési értékek problémáját, bár azzal itt nem is sokra mennénk, mivel a függvények nem nyúlhatnak az

adatbázisban tárolt adatokhoz (de attól még hiányoznak az összetett változótípusok). Ezen túl körülményes, hogy csak a kurzorokon keresztül lehet hozzányúlni ciklikusan az eredményekhez, a kurzorok viszont rendkívül buták. Nem lehet bármilyen eredményt hozzáadni, nem lehet később hozzáfűzni eredményeket, nem lehet benne pozicionálni, és még sorolhatnám. Szóval van még mit fejleszteni.

Vissza a témához: Triggerek

A *triggerek* olyan speciális tárolt eljárások, amelyek valamilyen esemény (`INSERT`, `UPDATE`, `DELETE`) esemény során hívódnak meg automatikusan. A trigger egy adott táblához van kötve, és az arra a táblára vonatkozó, előre megadott művelet során hajtódik végre.
Általában adatbázisba írás előtti ellenőrzésre, írás utáni számított érték kiszámítására, törlés előtti adatok bizonyos részeinek rögzítésére, stb. használhatjuk.
Megadható, hogy az adott esemény előtt (`BEFORE`), vagy után (`AFTER`) hajtódjon végre, egyébként egy szokványos tárolt eljárásról van szó, vagyis inkább függvényről... Eljárás, mert nincs visszatérési értéke, függvény, mert rengeteg olyan megkötés van, amit előzőleg a függvényeknél tapasztalhattunk: nem lehet másik tárolt eljárást, vagy függvényt meghívni, nem lehet tranzakciót kezdeményezni, jóváhagyni vagy visszavonni, de ami a legfontosabb: nem hivatkozhatunk közvetlenül más táblákra, de még arra a táblára sem, amihez a triggert rendeltük. Egyedül az aktuálisan érintett rekord értékeit kaphatjuk meg, vagy írhatjuk felül az alábbiak szerint: `INSERT` esetén a beillesztendő rekord oszlopai elérhetők a `NEW.oszlopnév` hivatkozással. Ebben az esetben az oszlopnév írható, azaz a `SET NEW.oszlopnév=érték` művelet lefut, ha megvan a megfelelő jogosultságunk. `DELETE` esetében a törlendő rekord oszlopai érhetők el az `OLD.oszlopnév` szintakszissal, de csak olvasható módon. `UPDATE` művelet esetén a felülírandó rekord értékeit az `OLD`, az új rekord értékeit pedig a `NEW` hivatkozáson keresztül lehet elérni (előbbi csak olvasható, az utóbbi írható).
Ezen kívül van még egy olyan – egyébként természetes – megkötés, hogy egy eseményhez egy adott időben (tehát például `BEFORE INSERT`) csak egyetlen trigger tartozhat.
Ha a fenti kivételeket leszámítjuk, akkor ugyanúgy használhatjuk, mint egy függvényt, élnek a vezérlési szerkezetek, vannak változóink, de nem láthatunk ki a függvényből.
Nézzünk egy példát trigger használatára:

```
CREATE TRIGGER ins_szorzo BEFORE INSERT ON termekek
    -> FOR EACH ROW SET NEW.fogyar = NEW.nagykerar
➥* 1.2;
```

Ez a trigger beszúrás előtt kiszámítja a fogyasztói árat a nagykereskedelmi ár felszorzásával, majd maga az `INSERT` művelet csak ez után fut le, tehát az érték bekerül a táblába – anélkül, hogy a „felhasználói” oldalról bármit csináltunk volna.
És még egy példa a *MySQL* dokumentációból, amely egy `UPDATE` eseményre levágja a megadott intervallumból kilógó értékeket. Íme:

```
mysql> delimiter //
mysql> CREATE TRIGGER upd_check BEFORE UPDATE ON
➥account
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.amount < 0 THEN
    ->         SET NEW.amount = 0;
    ->     ELSEIF NEW.amount > 100 THEN
    ->         SET NEW.amount = 100;
    ->     END IF;
    -> END//
mysql> delimiter ;
```

És ha már ide érkeztünk: hasznos újítás a `delimiter` parancs, amely arra szolgál, hogy megadjuk vele, hogy az `sql` parancs bevitele során (parancssoros kliens estén) milyen karakter zárja le magát a parancsot. Mivel az alapértelmezett határoló karakter a pontosvessző, ami egyben a tárolt rutinok nyelvének utasításait is lezárja, kénytelenek vagyunk átdefiniálni ezt addig, amíg begépeljük (bemásoljuk) a rutint. Jelen esetben a `//` karaktereket használjuk határolónak, majd mikor végeztünk, visszaállítjuk az alapértelmezett pontosvesszőre.
Sajnos a jelenlegi rengeteg korlátozás miatt még néhány kevésbé bonyolult dologra sem tudjuk felhasználni, és a dolgozni sem egyszerű vele.

Összegzés

Bár a fejlesztők törekvése helyes és tiszteletreméltó, még igencsak gyerekcipőben jár ez a dolog. A program készítői minden lehetséges ponton feltüntették, hogy a legtöbb lehetőség jelenleg is bővítés alatt áll, és ezt tudomásul véve is azt kell mondjam, hogy komoly dolgokra jelenleg (és még pár hónapig ez így is marad) nem alkalmas. Hiába a pehelysúlyú kategória zászlós hajója a *MySQL*, ezen projekt esetében ez nem számít, ugyanis a cikkben tárgyalt témakörök nem igazán sorolhatók ide, de ha figyelembe vesszük, hogy a tárolt rutinok kezelésénél soha nem fog a program a tökéletes sokoldalúságra törekedni, a tudása még akkor is gyenge. Hozzáteszem, az irány jól el van találva, csak még nem haladt az alkalmazás kellő számú lépést ezen az egyébként rögös úton. Nem beszélve az előbb már érintett problémáról, a pehelysúlyú megoldások alkalmazásáról egy inkább nehézsúlyú területen. Magam is kíváncsi vagyok, hogy a készítők hogyan fogják ezt a feladatot megoldani. Azt javaslom, várjunk még… érdemes próbálgatni, feltérképezni a jelenlegi fejlesztői változatot, és bár ezekre nem szoktunk alkalmazásokat építeni, feltételezem, hogy az első stabil változtok sem fogják tartalmazni a teljes fegyverarzenált, így ha valaki *MySQL*-ben szeretne magasröptű adattárolási megoldásokat tárolt rutinokkal kezelni, az talán várjon egy kicsit, s előbb néhány közepes erősségű probléma megoldására törekedjen. Majdnem biztos vagyok benne, hogy idővel ezen a területen is mélyen a szívünkbe lopja magát a *MySQL*.

Komáromi Zoltán

KAPCSOLÓDÓ CÍMEK

A MySQL honlapja: ➲ http://www.mysql.com
Magyar tükör: ➲ http://mysql.sote.hu

Számítógép hálózatok (16. rész)
Kapcsolatállapot alapú forgalomirányítás, hierarchikus forgalomirányítás

Folytatjuk tovább a dinamikus forgalomirányító eljárásokkal, most egy gyakorlatban is széles körben használt algoritmust mutatunk be: a kapcsolatállapot alapú forgalomirányítást. Ezek után megnézzük, mi a teendő akkor, ha hálózatunk olyan nagyra nőtt, hogy a forgalomirányító táblázatok már nem férnek el útválasztóink memóriájában.

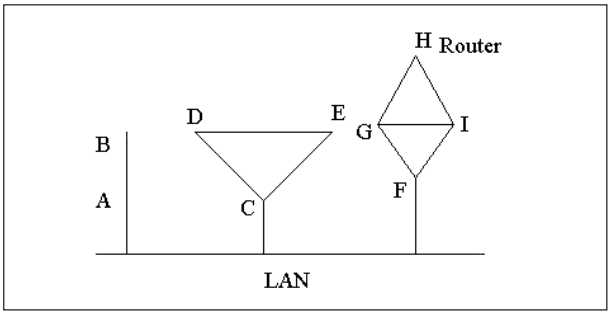
A távolságvektor alapú forgalomirányítás ugyan már egy dinamikus algoritmus volt, amit 1979-ig az internet elődje, az ARPANET is használt. A gyakorlatban azonban ez az eljárás megbukott. Leváltásában két dolog játszott szerepet. Először is az algoritmus döntéseinek meghozatalakor nem vette figyelembe az egyes vonalak sávszélességét. Ez a kezdetek kezdetén nem jelentett gondot, mivel az összes vonal 56 kb/s-os áteresztőképességgel rendelkezett, de miután egyes vonalak sávszélességét az eredetinek a többszörösére növelték, ez a hiányosság súlyos problémává kerekedett.

Az algoritmus bukásának valódi oka azonban nem ez volt, hiszen az eljárást könnyedén meg lehetett volna változtatni úgy, hogy figyelembe vegye az egyes vonalak közötti sávszélességbeli különbségeket is. Az igazi probléma inkább az volt, hogy az algoritmus a „rossz hírekre" továbbra is lassan reagált, és ezen még a megosztott látóhatár alkalmazása sem segített. Mivel a végtelenig számlálás problémájára nem sikerült elfogadható megoldást találni, egy teljesen új forgalomirányítási eljárást kellett kifejleszteni. Itt kezdődik a ***kapcsolatállapot alapú forgalomirányítás*** története.

Az algoritmus működése egyszerű: az útválasztók először a szomszédaikkal ismerkednek meg, majd megmérik a közöttük lévő késleltetéseket. Az így kapott információkat az alhálózat többi útválasztójának is elküldik, majd Dijkstra algoritmusának segítségével kiszámítják az alhálózat összes pontjához a legrövidebb utat.

Ismerkedés a szomszédokkal

Az útválasztó első feladata tehát a szomszédság feltérképezése. Ebben segít neki egy speciális, úgynevezett ***HELLO*** csomag, amelyre minden útválasztó válaszol, és a válaszban elküldi saját egyedi azonosítóját. (Az azonosítók egyedisége nagyon fontos. Tegyük fel, hogy egy útválasztó azt látja, hogy két másik útválasztó is szomszédja az X című útválasztónak. Ha az azonosítók nem egyértelműek, akkor nem lehet eldönteni, hogy a két útválasztó ugyanannak az útválasztónak a szomszédja-e, vagy két különböző X azonosítójú útválasztó van az alhálózatban).

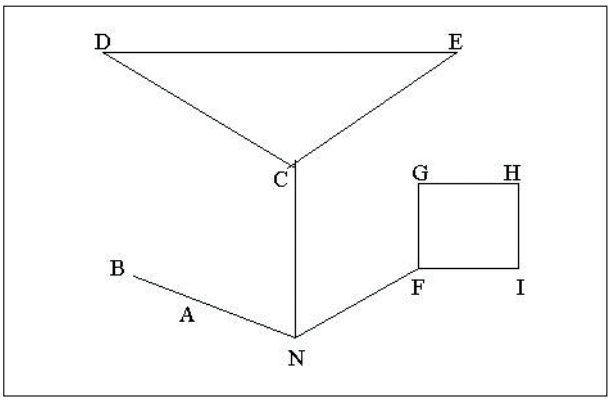


1. ábra Három router egy LAN-on

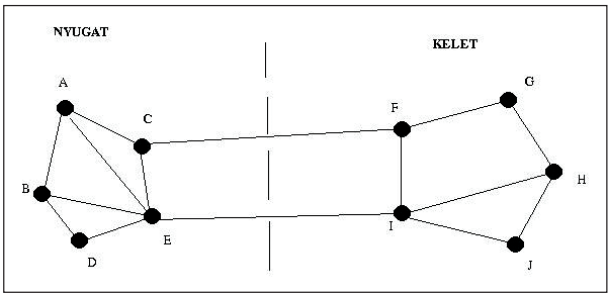
Amikor egy útválasztót bekapcsolunk, akkor az „becsönget" minden szomszédjához, azaz kiküld egy HELLO csomagot az összes kimenetén, majd vár a válaszok megérkezésére. Felmerül a kérdés, mi történik akkor, amikor egy kimeneten keresztül több útválasztót is el lehet érni, például két vagy több útválasztó egy LAN-ra van felfűzve. Az *1. ábrán* egy ilyen topológiát láthatunk, ahol A, C és F útválasztó közvetlenül, egy LAN-on keresztül kapcsolódnak egymáshoz. Ilyen esetekben célszerű a LAN-t helyettesíteni egy „virtuális" csomóponttal. A *2. ábrán* a LAN-t kicseréltük egy N címkéjű pontra. Ezek után például az A-ból a C-be az ANC úton keresztül juthatunk el.

Késleltetések mérése

A szomszédokat nem elég megismerni, hanem azt is tudni kell, hogy mekkora a – szomszédokhoz vezető – vonalak késleltetése (vagy költsége). Erre a feladatra is egy speciális csomagot alkalmaznak, amelynek neve ***ECHO*** csomag. Amikor egy útválasztó egy ilyen csomagot kap, azt soron kívül, azonnal visszaküldi azon a porton, amelyiken beérkezett. Miután az útválasztó kibocsátott egy ECHO csomagot,



2. ábra A LAN-t úgy is felfoghatjuk, mint az alhálózat egy csomópontját

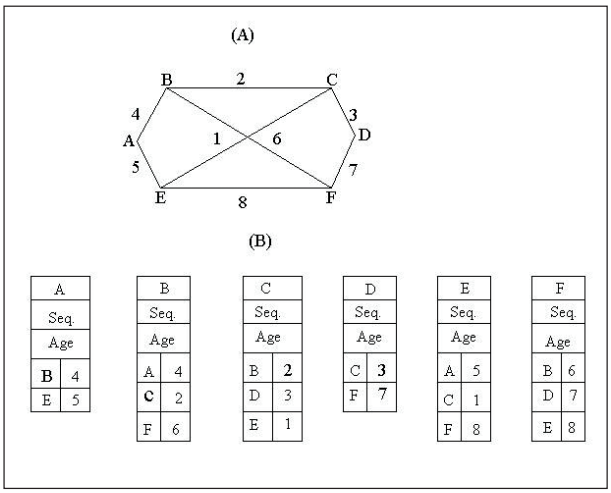


3. ábra Az ehhez hasonló toplógiájú hálózatokban nem feltétlenül előnyös, ha a vonalak költségeinek kiszámolásakor a terhelést is figyelembe vesszük

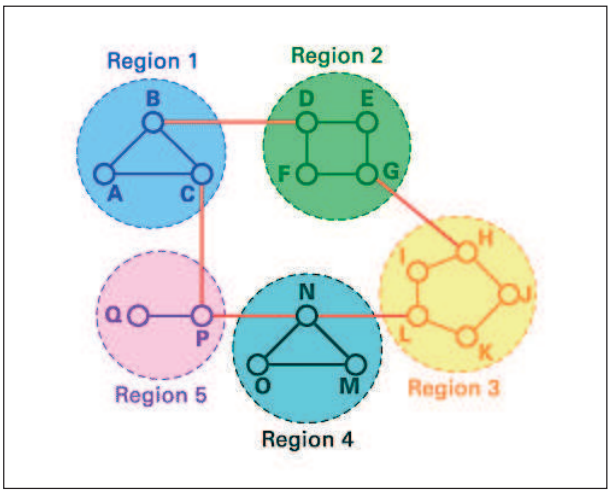
megméri, mennyi időbe telik, míg visszaér hozzá. Ha ezt az időt osztjuk kettővel, akkor egy becslést kaphatunk az útválasztó és a szomszédja között lévő vonal késleltetésére.
A pontosabb becslés érdekében ezt a mérést időnként érdemes megismételni. Ilyenkor a becsült költség mindig a mérések eredményeinek az átlaga.
Ha a vonal késleltetésének becslésekor figyelembe szeretnénk venni a terhelést is, akkor a mérés menete annyiban különbözik az előbbiektől, hogy az útválasztó az ***ECHO*** csomagot a várakozási sor végére teszi (ahol az útválasztón átmenő csomagok is várakoznak), és a „stoppert” onnantól indítja (nem pedig attól az időponttól, amikor az ***ECHO*** csomag távozik az egyik kimeneten).
Látszólag értelmetlennek tűnik az a kérdés, hogy figyelembe vegyük-e a terhelést, vagy sem. Mondhatnánk, hogy persze, vegyük figyelembe, hiszen ha van két ugyanolyan sávszélességű vonal, ahol az egyik teljesen leterhelt, a másikon pedig éppen semmi sem csordogál, akkor az utóbbi legyen része a kijelölt útnak. A logikus választás valóban ez lenne, hiszen így a csomag jóval hamarabb célba érhet.
Vannak azonban helyzetek, amikor kifejezetten hátrányos, ha a késleltetésbe a terhelést is beszámítjuk. Vessünk egy pillantást a *3. ábrára*! Láthatjuk, hogy az alhálózat nyugati részét a keleti résszel a CF és az EI vonal köti össze. Tegyük fel, hogy a legtöbb kelet felé menő csomagot a CF vonalon keresztül irányítjuk, így ott a terhelés sokkal nagyobb lesz, mint az EI-n. Ha a becsléskor a vonal terhelését is megvizsgáljuk, akkor azt fogjuk kapni, hogy a csomagok EI-n keresztül kisebb késleltetéssel jutnak célba. Ezek után a kelet felé tartó forgalom oroszlánrésze az EI-n kerül továbbításra, így az válik terhelté, míg a CF-en a forgalom ritkul. Ne csodálkozzunk, ha a következő becslés eredményeképpen a CF lesz része a legrövidebb utaknak.
A vonalak késleltetésének folyamatos változása miatt az útválasztók forgalomirányító táblázatai is állandóan módosulnak, méghozzá túl gyorsan. Ez nem túlzottan szerencsés dolog, ugyanis a forgalomirányítás nem lesz megbízható, és rengeteg nem várt problémával is szembesülhetünk. Ilyen esetekben tehát nem jó, ha a terhelést is nézzük az egyes vonalakon. Ehelyett megtehetjük például azt, hogy a forgalmat testvériesen megosztjuk a két vonal között, habár így nem biztos, hogy minden csomag a lehető legrövidebb úton fog célba jutni.

Kapcsolatállapot csomagok

Eddig sok újdonságról nem volt szó, a fentiekkel már találkozhattunk valamilyen formában a távolságvektor alapú forgalomirányításnál is. A kapcsolatállapot alapú forgalomirányítás lelke azonban az úgynevezett kapcsolatállapot csomagok, amelyek segítségével az útválasztó megoszthatja társaival az általa összegyűjtött adatokat. Ezek a csomagok tartalmazzák egyrészt a feladót, annak életkorát (lásd később), illetve annak szomszédait, és az azokhoz tartozó késleltetéseket. Ezeket az információkat az alhálózat összes útválasztója kicseréli egymással. A *4. ábrán* láthatunk példát az alhálózaton terjedő kapcsolatállapot csomagokra.
Nem egyszerű az a kérdés, hogy egy útválasztónak mikor érdemes összeállítani és szétküldenie a kapcsolatállapot csomagjait. Általában kétféle megközelítést alkalmaznak.
Az első esetben az útválasztók periodikusan, azaz szabályos időközönként küldik szét az általuk összegyűjtött adatokat.
A másik lehetőség az, hogy a kapcsolatállapot csomagok csak bizonyos események bekövetkeztében jönnek létre, például ha egy vonal megszakad, vagy egy szomszéd útválasztó elhalálozik, illetve megjavul.
A kapcsolatállapot csomagokat az útválasztók az elárasztás módszerével szórják szét az alhálózaton, tehát az átmenő kapcsolatállapot csomagokat az útválasztók az összes kimenetükön továbbküldik (kivéve azt a portot, amelyikről a csomag beérkezett). Mivel ez a módszer exponenciális léptékben gyártja a csomagok duplikátumait, a folyamatot valamiképp kordában kell tartani. Az előző részben bemutattunk erre egy megoldást, miszerint minden csomagnak kell rendelkeznie egy sorszámmal, amely minden csomagküldés alkalmával eggyel nő. Minden útválasztó számon tatja, hogy melyik portról milyen sorszámú csomagok érkeztek eddig. Ha egy olyan csomag jön, amelynek a sorszáma nincs ezen a listán, akkor azt az útválasztó elárasztja, és a sorszámát feljegyzi. A listán szereplő sorszámú csomagok viszont többé nem kerülnek elárasztásra. Az ilyen csomagokat az útválasztók egyből ki is dobják.
Ez ugyan egy jól használható módszer, mégis akadnak vele problémák. Az első és legkézenfekvőbb dolog az, hogy ha a sorszámok egyszer csak átcsordulnak, akkor az egész rendszer dugába dől. (Ha például egy bájttal ábrázolnánk a csomagok sorszámait, akkor a 255. sorszám után a 0 lenne a következő, amelyet egy útválasztó sem fog elárasztani). Erre igazából csak egy megoldás létezik, mégpedig az, hogy a sorszámokat minél több biten ábrázoljuk. Persze a probléma még így is fennáll, viszont például 32 bit esetén a sor-



4. ábra Kapcsolatállapot csomagok



5. ábra Kétszintű hierarchikus forgalomirányítás

számok csak 137 év múlva fordulnának át. (Feltéve ha az útválasztók átlagosan másodpercenként indítanak kapcsolatállapot csomagokat).
Persze az útválasztók nem arról híresek, hogy matuzsálemi korokat éljenek meg, előbb-utóbb mindegyik elromlik, kifagy vagy egyéb katasztrofális eseményekkel kell szembesülnie. Útválasztóink halandósága újabb problémákat szül: elfelejtik, hogy milyen sorszámú kapcsolatállapot csomagokat bocsátott már ki. Ha a sorszámozást a nullánál kezdi, akkor valószínű, hogy csomagjai nem jutnak messzire, hiszen a többi útválasztó elavultnak fogja tekinteni őket.
Az is problémát rejt, hogy az útválasztók lista helyett csak egy számlálót tartanak karban. A számláló aktuális értéke alatti sorszámok jelölik az elavult csomagokat. Ha egy nagyobb sorszámú csomag érkezik, az útválasztó azt elárasztja, majd a számláló értékét a csomag sorszámára növeli. Ez egy rendkívül memóriatakarékos módszer, viszont a bithibákra nagyon érzékeny. Ha ugyanis a sorszámban csak egy bit is megsérül, akár nagyságrendekkel nagyobb számot is kaphatunk. Például a 2-es és a 65538-as sorszám bináris alakja közötti eltérés csupán egy bit. Ha az útválasztó egy ilyen csomagot kapna, akkor 2-től 65538-ig minden csomagot el fog dobni. A probléma megoldásához szükséges, hogy a csomagba a sorszám mellé egy életkor mezőt is definiáljunk, amelynek értéke másodpercenként eggyel csökken. A csomagok életkorát az útválasztók is csökkentik elárasztáskor. Amikor egy csomag életkora eléri a nullát, akkor megsemmisül.
A gyakorlatban a beérkező kapcsolatállapot csomagok nem kerülnek egyből továbbításra, hanem először egy pufferbe kerülnek. Ha ugyanattól a forrásból még egy kapcsolatállapot csomag érkezik, akkor az útválasztó összehasonlítja a két csomag sorszámát. Ha ezek megegyeznek, akkor a második csomag már egy duplikátum, így az eldobható. Ha a sorszám különbözik, akkor a régebben érkezett csomag által hordozott információ már nem érvényes, így azt kell kidobni.
Hogy az algoritmus még robosztusabb legyen – azaz ellenállóbb legyen a nemvárt vonalhibákkal szemben – az útválasztók a megkapott kapcsolatállapot csomagokat nyugtázzák egymásközött. Hasonlóan a kapcsolatállapot csomagokhoz, a nyugták sem kerülnek azonnal elküldésre, először ők is a pufferben várakoznak.

A pufferben a csomagok mellett a forrásuk, a sorszámuk, a koruk és a küldési, illetve a nyugtázási jelzőik is tárolva vannak. Ezek közül az utóbbi kettő szorul még némi magyarázatra. A küldési jelző azt mondja meg, hogy a kérdéses csomagot mely kimeneteken kell továbbítani, a nyugtázási jelző pedig azt határozza meg, hogy a nyugtát kinek kell visszaküldeni.
Mit nyerünk ezzel? Tegyük fel, hogy a 4. ábrán feltüntetett alhálózati topológiában a B útválasztó kétszer is az E útválasztó kapcsolatállapot csomagot. Például először az EAB majd az EFB útvonalon keresztül. Ebben az esetben a csomagot felesleges minden porton elárasztani, elegendő csak a C felé vezető kimeneten. Így az útválasztó a pufferben átírja a csomag küldési jelzőjét úgy, hogy csak a C felé továbbítódjon (a nyugtát viszont A-nak és F-nek is vissza kell küldeni).
Ezzel a technikával csökkenthetjük a kapcsolatállapot csomagok által előidézett forgalomnövekedést.

Az új útvonal kiszámítása

Miután az útválasztók kicserélték információikat, felépíthetik az új forgalomirányító táblázatot. A legrövidebb utak kiszámítása a Dijkstra algoritmus segítségével történik.
Fontos megjegyeznünk, hogy a kapcsolatállapot csomagok révén megkapott információ mérete jóval meghaladja a forgalomirányító táblázat méretét. Gondoljuk csak meg: ha az alhálózatban N darab útválasztó van, és minden útválasztónak K darab szomszédja van, akkor a beérkező adatok tárolásához legalább K*N méretű memóriára van szükség. Ha nagyon nagy a hálózat, akkor ez komoly gond, mivel nem áll rendelkezésre elegendő memória (sőt, az igazán nagy hálózatok esetében a teljes forgalomirányító táblázat sem fér el az útválasztó memóriájában). Nem is beszélve a hálózat méretével arányosan növekvő számításiigényről, amely az új táblázat kiszámításához szükséges.
Ennek ellenére a kapcsolatállapot alapú forgalomirányítást a mai számítógép hálózatokban is széles körben használják. A későbbiekben részletesen megismerkedünk olyan protokollokkal, amelyek erre a forgalomirányítási eljárásra épülnie (például az OSPF protokoll).

Cél	Vonal	Ugrások
A	–	–
B	B	1
C	C	1
D	B	2
E	B	3
F	B	3
G	B	4
H	B	5
I	C	5
J	C	6
K	C	5
L	C	4
M	C	4
N	C	3
O	C	4
P	C	2
Q	C	2

6/a. ábra Az „A” útválasztó táblázata hierarchikus forgalomirányítás nélkül

Hierarchikus forgalomirányítás

Az imént említettük, hogy az alhálózat méretének növekedése maga után vonja a útválasztók forgalomirányító táblázatainak növekedését is. Előbb-utóbb alhálózatunk akkorára duzzadhat, hogy nem lesz olyan útválasztó, amely elegendő memóriával rendelkezne ahhoz, hogy az összes kapcsolóelemhez külön bejegyzést rendelhessen. Ilyenkor a forgalomirányítást ***hierarchikusan*** kell végeznünk, hasonlóan, mint a távbeszélőhálózatok esetében.

A hierarchikus forgalomirányítás esetén az útválasztók tartományokba vannak csoportosítva. Minden útválasztó csak a saját tartományába tartozó kapcsolóelemeket ismeri, a többi régió topológiája rejtve marad számára. Az *5. ábrán* láthatunk egy példát az alhálózat hierarchikus felépítésére. Az alhálózat összesen 17 kapcsolóelemet tartalmaz, ami azt jelenti, hogy ha nem lenne a forgalomirányítás hierarchikusan szervezett, akkor minden útválasztónak egy 17 bejegyzést tartalmazó útvonalválasztó táblázatot kéne észben tartania *(6/a ábra)*. Ellenkező esetben a tartományokat egy-egy csomópontba „sűríthetjük”, így az útválasztóknak a lokális kapcsolóelemeken kívül csak tartományonként egy-egy bejegyzést kell tárolniuk.

A *6. ábrán* láthatunk példát az első tartományban lakó Az útválasztó forgalomirányító táblázatára. Az első esetben nincs hierarchikus forgalomirányítás, így a táblázat 17 sort tartalmaz. A második esetben láthatjuk, hogy a második tartomány felé igyekvő csomagok mindig a BD vonalon keresztül haladnak, míg a többi régió felé a CP vonal vezet.

Cél	Vonal	Ugrások
A	–	–
B	B	1
C	C	1
2. tartomány	B	2
3. tartomány	C	4
4.tartomány	C	3
5. tartomány	C	2

6/b. ábra Az „A” útválasztó táblázata hierarchikus forgalomirányításnál

A táblázatok méretkülönbsége szembeötlő, a táblázat a hierarchikus forgalomirányítás esetében 17 helyett csupán 7 bejegyzést tartalmaz. Általánosan elmondható, hogy minél nagyobb a tartományok száma a kapcsolóelemek számához viszonyítva, annál kisebb méretűek lesznek az útválasztók forgalomirányító táblázatai.

Mivel semmi sincs ingyen, a helyspórolásnak is megvan a maga ára: le kell mondanunk arról, hogy a csomagok minden esetben a lehető legrövidebb úton keresztül jutnak célba. Maradva az 5. ábrán látható példánál, ha az Az útválasztótól a harmadik tartományban található H útválasztóhoz szeretnénk eljutni, akkor kicsit kerülni fogunk. Ugyan leghamarabb a második régión keresztül juthatnánk el, az A útválasztó mégis az ötödik tartomány felé irányít minket. Miért? A válasz egyszerű: lehet, hogy speciálisan a H útválasztó felé közelebb lenne, ha a BD vonalon indulnánk el, de a harmadik tartomány legtöbb útválasztója mégis a CP vonalon keresztül érhetőek el a leghamarabb. (Megmutatható azonban, hogy a hierarchikus forgalomirányításból adódó többletutak általában nem számottevőek, legalábbis bőven az elfogadható határon belül vannak. Persze kivételek mindig vannak).

Az 5. ábrán egy kétszintű hierarchiára láthattunk példát. Az igazán nagy hálózatok esetén azonban ez sem segít a dolgon, a forgalomirányító táblázatok mérete továbbra is elfogadhatatlanul nagyok maradnak. A tartományokat így tovább kell bontanunk zónákra, a zónákat kerületekre, a kerületeket csoportokra, és így tovább.

De vajon hány hierarchiai szintet érdemes definiálni? Ennek kiszámításához ismert egy összefüggés, amely szerint az optimális hierarchiai szintek száma ln N, ahol N az alhálózatban lévő útválasztók száma. Ebben az esetben minden útválasztónak e * ln N számú bejegyzést kell tárolnia a memóriájában.

A következő részben továbbra is a forgalomirányítással foglalkozunk, de most már azt vizsgáljuk, mi a helyzet, ha a gépek nem otthonülő életmódot folytatnak, hanem összevissza mozognak. Ezenkívül szó lesz még a többesküldéses, illetve az adatszórásos forgalomirányításról.

Garzó András
garzo@interware.hu

Szoftverjogi csiki-csuki, avagy a szoftver analizálás és annak határai

A szoftverek visszafejthetőek. Ezzel vélhetően nem árultam el nagy újdonságot. Azzal talán annál inkább, hogy a szoftverek visszafejtése bizonyos körülmények között jogszerű is lehet.[1]

A szoftver visszafejtése

A szerző engedélye nélkül is jogosult a program használatára licencben vagy más módon jogosított felhasználó, vagy megbízottja (hiszen egy mezei felhasználó az esetek 99,99%-ában erre úgysem lenne képes) vagy egy programozó akár munkaköri kötelességként is a kód többszörözésére, szükség esetén (vissza)fordítására egy speciális információhoz való hozzájutás érdekében. Ezen információ kizárólagosan azon célt szolgálhatja, hogy a visszafejtett szoftvert más szoftverekkel együtt tudja működtetni a felhasználó. A központi fogalom tehát az „interoperabilitás" megteremtése.

A reverse engineering

Az alábbi bekezdést legjobb lesz, ha a profik átugorják, ez tartalmazza ugyanis annak összefoglalóját, hogy mit ért egy jogász a ***reverse engineering*** fogalma alatt.[2]
Azon eljárást, melynek keretében a tárgykódból különféle, a továbblépéshez szükséges információkat nyernek, revese engineeringnek nevezzük. Ennek egyik eszköze a dekompiláció, ami a tárgykódnak egy fordítóprogram segítségével forráskódba fordítását jelenti. Ám – lévén, hogy rengeteg fordító (kompiláló) program létezik, – amennyiben a visszafordításhoz nem jó dekompiláló programot (azaz nem az eredeti kódolásnak megfelelőt) használunk, a kívánt információk helyett egy értelmetlen és értelmezhetetlen karaktersorhoz jutunk. Ezért általában, (ha a fordító program „ujjlenyomata" (fingerprint) alapján nem képes a fordítást végző azonosítani a programot) akkor egy alap programozási nyelvre, assembly-be fordítja. Ezen lehetőség bármiféle tárgykód esetében fennáll, ám azzal a kockázattal, hogy a szerző kommentár sorait nem nyerjük vissza, és az assembly csak szakember számára olvasható programnyelv.
A kompilált sorokat aztán az esetleges változtatásokat követően természetesen lehet rekompilálni, amely eljárást követően újra tárgykódban lévő programhoz jutunk.

A de-, rekompiláció lényegi elemei tehát egyrészt a fordítás, másrészt pedig egy speciális többszörözés. Ezen eljáráshoz való hozzájárulás tehát a szerzőknek általában nem is áll érdekükben. Ezért csak különböző előfeltételek együttállása esetében lehet egyáltalán szó ezen eljárások jogszerű alkalmazásáról. Lássuk most ezeket.

Az információhoz való alternatív hozzájutás hiánya

Amennyiben a megkívánt információkhoz szabad hozzájutást biztosít a szerző, a programot nem kell felfejteni. Erre példa a GPL licencelésű a szabad szoftverek esete, hiszen itt kötelező jelleggel megadják a kódot magát, vagy azt elérhetővé teszik az Interneten
Felmerülhet azonban kérdésként, hogy milyen tágan értelmezhető „a megkívánt információt „könnyedén megismerhetősége", mint kitétel. Ide sorolja a jogirodalom a régebben nyilvánosságra hozott, ám kis kutatással fellelhető információkat. Ellenpéldaként viszont ide tartozhat, ha az az információt további ellenszolgáltatásért cserébe kínálják fel. Nem szabad elfelejteni, hogy a könnyen hozzáférhetőség mindig egyedi mérlegelést megkívánó fogalom.
Ugyancsak kérdés, hogy kinek is áll érdekében az információhoz jutás biztosítása, elvárható-e a felhasználótól, hogy a fejlesztő felkeresésével időt töltsön, annak érdekében, hogy a dekompiláció ne minősüljön joggal való visszaélésnek? Valószínűleg, rövid határidő tűzés mellett a fejlesztő felkeresése és felszólítása az adat szolgáltatására jogos elvárás.

Az információk célhoz kötött volta

A másik alapvető feltétel, hogy a megszerzett információk ténylegesen a programok együttműködtésének lehetőségét biztosítsák. Minden más céllal történő visszafejtés engedélyköteles. Ugyanúgy nem szabad ötletek megszerzése céljából dekompilálni, mint ahogy jogsér-

[1] Szjt. 60. §.
[2] Kritikai észrevételeket ide várok: agi@abend.hu

tés bizonyítása sem megengedett[3] ezen a módon. (Amennyiben bíróság rendeli el a szakértő általi dekompilációt, az persze más eset, ám a bíróság jogosult a sokkal egyszerűbb utat is választani, kérni a forráskód kiadását.) Sőt a felhasználó azt sem teheti meg, hogy a szoftver hibáját ezúton bizonyítja, hiszen – bár a cél nemes – mégsem felel meg az irányelv és a törvény szövegének, mely egyértelműen csak „önállóan megalkotott szoftver más szoftverekkel való együttes működtetéséhez szükséges információ megszerzés érdekében" teszi lehetővé az eljárást. A visszafejtés nem használható sem a felhasználó érdekeit közvetlenül szolgáló testre szabás céljából, sőt még kutatási célokból sem.

Érdekes azonban, hogy a kompatibilissé tétel abban az esetben is engedélyezett, ha egy konkurens programmal akarják az adott szoftvert együttműködésre bírni.

Ugyancsak speciális eset a hardverrel való együttműködés, együttműködtetés, hiszen a szöveg szó szerint csak szoftverekhez való illesztést engedi. A mai modern hardver elemek lehetővé teszik a szoftverek hardveres implementálását, azaz szoftver feladatok hardverben történő ellátását. Ez pedig indokolttá tenné a dekompilálás engedélyezését erre az esetre is. Hiszen ennek elmaradása a piac megosztásához vezethet, ami versenyjogi aggályokat vet fel.

A szoftvergyártók általában ellátják terméküket egy interface specifikációval, mely esetenként azonban nem teljes körű, így a visszafejtés elkerülhetetlen. Ám az abból nyert információk (akár az interface alkotó kódja is) a visszafejtő rendelkezésére állnak. Ezek közül azonban a szabályozás alapelvének megfelelően elsődlegesen a specifikációt kell kiegészíteni és azt használni, nem pedig a konkrét kódot átemelni.

A személyi kör

A dekompilálás három személyt érinthet. A felhasználási szerződésben jogosítottat, vagy a szoftver felhasználására jogosult más személyt (pl.: munkaviszony alapján eljárót) illetve ezek megbízottját, aki semmiféle közvetlen viszonyban nem áll a szerzővel. Ezen személy eljárási jogosultságát elég egy polgári jogi jogviszonnyal igazolni. A személyi kör utolsó elemmel való kibővítését az tette szükségessé, hogy a dekompilálás – mint már a bevezetésben említésre került – speciális tudást igényel.

Az információ terjedelme

A szükséges programrészek jelentik az információ megismerésének határát. Feltétlen megismerést igénylő, tehát az együttműködéshez szükséges információt tartalmaznak például a (hivatalos és nem hivatalos[4]) csatlakozó felületek (interface).

A dekompilációt végző személytől elvárható, hogy a kézikönyvek és a rendelkezésre álló irodalom információi alapján kiderítse, hogy melyik programrészt kell dekompilálni. Egyes – az újabb programokra egyre jellemzőbb – megoldások esetében, melyek könyvtárak bevonásával dolgoznak ez a hely könnyebben meghatározható, azaz nem kell valamennyi könyvtárat visszafordítani. Más esetekben viszont, mikor szorosan összefüggő programszövegről van szó, felmerülhet a teljes fordítás elkerülhetetlensége. A vizsgálat terjedelmi korlátait mindig speciális vizsgálat keretében kell ellenőrizni, vizsgálni kell, hogy adott szakértelem mellett előre látta-e vagy láthatta-e, hogy a program mekkora részének visszafordítása szükséges. Azonban még annak sincs kialakult gyakorlata, hogy a terjedelmi korlátok „túl nem lépését" ki köteles bizonyítani.

A megismert információ szabadsága

A szerző érdekeinek védelme szükségesé teszi, hogy a visszafordítás keretében elnyert információ ne legyen továbbadható. E védelmi szint még abban az esetben is fenntartandó, ha más felhasználók ugyancsak visszafordításokra lesznek kénytelenek, melyek végeredményeként természetesen ugyanazon eredményeket kapják. Az említett korlátozás olyannyira kereteket jelent, hogy a megtalált interfész nem adható közre a szakirodalomban, viszont az érem másik oldalaként az általunk írt szoftver kézikönyvében megjelenhet, hogy hogyan tehető kompatibilissé adott esetben egy operációs rendszerrel a programunk. Egyedi megoldás, hogy ha a szerzői oldalról visszaélést tapasztalnak, vagy piacot uraló helyzet áll fenn, a kartelljogi szabályokra hivatkozva követelhetjük a lényeges információk közreadását.

Az utánozási tilalom

Ismételt tilalomként jelenik meg, a nyomatékosság kedvéért, hogy a megszerzett ismeretek alapján nem lehet egy az eredetihez hasonló programot létrehozni, sem olyat, mely hasonló elvekre épülne, vagy hasonló funkciókkal lenne ellátva. Ez voltaképpen versenyjogi kérdésnek tekinthető. Kétségtelenül tisztességtelen piaci magatartást testesítene meg a fél, aki az említett módon hozzájutva az információkhoz kezdi meg konkurens szoftver terjesztését.

Ez azonban nem jelenti azt, hogy ne lehetne – az eredeti használata nélkül alkotni egy másikat, hasonló céllal, hiszen ezen tilalom léte teljesen monopolizálttá tenné a piacot... egyetlen operációs rendszer működne, amin egyetlen szótárprogramot és egyetlen könyvelő programot lehetne futtatni, de mivel az ötletek nem védettek, így lehetőséget kell biztosítani másnak is hogy egy adott probléma megoldására szoftvert fejlesszen.

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.

[3] Dreier: GRUR 1993 781-785

[4] Ezen megkülönböztetés versenyjogilag lehet releváns, hiszen a nem hivatalos interface-ek arra szolgálnak, hogy azokat csak maga a gyártó használhassa.