Itt volt már az ideje!

Egyre többször fordul elő, hogy a GNU/Linux visszaköszön rám különböző munkahelyeken. Ez önmagában nem annyira meglepő, de ha hozzáteszem, hogy „mint asztali munkaállomás”, akkor mindjárt érthető, miért lelkesedem ezen oly nagyon. Úgy látszik, a linuxos világ feléledt, megrázta magát és sebességet váltott. Olyan fontos és hihetetlenül bosszantó gondok oldódtak meg, mint például az X-felület alatti betűtípus-kezelés (az X végre közvetlenül támogatja a FreeType betűkészleteket, lásd a 72. oldalon kezdődő cikket ***Tóth Béla*** tollából), a régen várt helyesírás-ellenőrző (➲ www.szofi.hu/gnu/magyarispell), az irodai csomagok (új AbiWord, az OpenOffice.org RC3 stb.) és még sorolhatnám.

Ami szívemnek különösen kedves a GNU/Linux világában, hogy olyan régi kedvenceket (vagy éppen átkokat) tesznek elérhetővé, mint a C64-es alatti játékok (lásd a 46. oldalon lévő cikket) vagy a jó öreg Clipper egy „ikertestvérét”, a clip csomagot (56. oldal). Ez a csomag azoknak lehet különösen szívhez szóló, akik – hozzám hasonlóan – eme különösen csábos programnyelv segítségével készítettek komoly és megbízható (hmm…) rendszereket. Egy rövid történetet, ha megengedtek: egyik nagyon jó barátom mesélte, hogy évekig dolgozott Clipperben, mígnem egyszer egyik (akkor már több éve tökéletesen működő) rendszerének üzemeltetői küldtek egy hibajelentést: kinyomtatták a képernyőt, majd átfaxolták. A faxon két szó szerepelt: Syntax Error.

A védelemről is szólunk. Most két témakörrel foglalkozunk kiemelten, a hálózati védelem „egygépes” változatáról, a személyes tűzfalakról (***Deim Ágoston***, 66. oldal), illetve egy rendkívül bosszantó jelenség elleni titáni küzdelem kiemelkedő eszközéről, a levélszemetet nyakon csípő SpamAssassin csomagról (***Varga S. Csaba***, 64. oldal). A helyi tűzfal ötlete és célja, hogy gyorsan és könnyedén fel tudunk állítani szűrőrendszerünket, az Ágoston által bemutatott programok pedig igen jó szolgálatot tesznek, ha a felhasználó egy lusta malac, és nincs kedve megtanulni a tűzfal kézzel történő beállítását.

A SpamAssassin pedig tényleg nagyon hasznos tud lenni. Teljesen véletlen volt, hogy mind Csaba, mind ***David A. Bandel*** (75. oldal) megemlíti. Én már egy kicsit ideges kezdek lenni, hogy az Interneten nem lehet nyugodtan dolgozni, az embert mindenképpen megtalálják. Nálunk a levelezés két rétegre van bontva, mindkét rétegen ellenőrzés folyik (jogosultságok, vírus, levélszemét), a levélkiszolgálót üzemeltető munkatársam éppen a múltkor fejtette ki, hogy döbbenetesen sok mindenre szűrünk. Jelenleg naponta csupán három-négy szemét jön át, plusz egy-két vírus, ha pár napig kihagyjuk az adatbázisok frissítését. Kész dili. Még szerencse, hogy van, aki igyekszik a levélkiszolgálókat felügyelő szakembereket is szórakoztatni. A minap jelent meg egy felhívás, többek között az User Friendly oldalon (lásd a keretes részt). Természetesen nem hagyjuk abba az oly sok vitát kavaró magyarítások témakörét, úgy érzem, megérett a helyzet egy külön rovatra, ennek beköszöntőjét találjátok a 32. oldalon. És hogy (szerencsére) nem csak mi érezzük fontosnak a magyarul beszélő rendszereket, jól bizonyítja, hogy végre az LME berkeiben is megalakult a Honosítás csoport. Az új csoport elnökével, ***Fejős Tamással*** beszélgettünk (lásd a 21. oldalt). A csoport tele van reményekkel és tervekkel, remélem, tényleg hatékonyan képesek majd működni.

De ha már idehaza járunk, egy különlegességre is fel szeretném hívni a figyelmeteket. Amerikai társlapunkban rendszeresen cikkezik egy helyi jogász, Lawrence Rosen, aki mostani számunkban az UCITA-ról ír. Hogy ez ne legyen elég, ***Granek István*** mutatkozik be az újság hasábjain pár oldal erejéig, és bevezet minket az itthon is kevésbé szövevényes szabályozások egyik legfrissebbikébe, a számítástechnikai rendszer és adatok elleni bűncselekmények világába.

Van, hogy úgy kell összeállítsunk GNU/Linuxot, hogy kevésbé hozzáértők is könnyen használhassák, vagy egyszerűen szeretnénk elérni, hogy a grafikus felület alatt használhatóak legyenek a betűkészletek, ne azzal bosszantsa az ember magát, hogy miért nem tudunk filmet nézni mindenféle programozói tudás nélkül az X alatt stb. Ez egy sokszor visszaköszönő témakörhöz vezet, az X beállításához. E művelet egyik legfontosabb lépése az XF86Config fájl céljaink és igényeink szerinti átalakítása. Ha az önműködő telepítőprogramok nem képesek megbirkózni e feladattal, ott a lehetőség, hogy mi magunk írjuk át a fájlt. Ehhez nyújt egy-két támpontot Tóth Béla a 72. oldalon. Egy dolgot emelnék ki, amivel magam is sokat küzdöttem (hogy finoman fogalmazzak), ez pedig a betűkészletek kérdése. Az új 4.x-es X-nél végre megjött a fejlesztők esze, és készítettek egy modult (freetype), mely lényegében megoldja ezt a kérdéskört. Nem kell végre szenvedni az aluldokumentált és támogatástól igencsak kellemesen mentes xfstt és xfs-xtt csomagokkal! No, azért ne olyan hevesen… Nem lehet csak úgy egyszerűen megkönnyíteni a rendszergazdák életét! Elkészült egy xtt modul is, csak hogy bosszantsanak minket. Mindenesetre végre épeszűen lehet dolgozni magyar szöveggel a grafikus felület alatt, végre remény csillant: küzdhetünk, hogy az X, a gs, az AbiWord és az OpenOffice.org alatt is meglegyen mind.

Mindenkinek kellemes olvasást és jó linuxozást kívánok!

Azt hiszed, hogy kézzel ki tudod szűrni a levélszemetet?

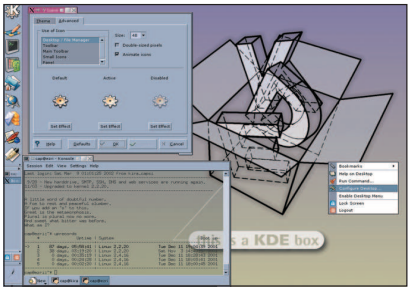
Itt az alkalom, hogy bebizonyítsd! A BVRP Software támogatásával új játék indul SPOOL néven. Az alapötlet, hogy levélkiszolgálót üzemeltetsz (rémálom), és kénytelen vagy folyamatosan növekvő hálózatodat megvédeni a levélszeméttől és a vírusoktól ? sajátkezűleg!…

Szy György
a Linuxvilág főszerkesztője, a Kiskapu Kiadó vezetője. Mindenki véleményét és levelét örömmel várja az alábbi levélcímen: Szy.Gyorgy@linuxvilag.hu

Programvadászat

KDE 3.0

Kiadták a KDE-projekt által fejlesztett és karbantartott grafikus felület 3.0-s változatát. A KDE-projekt világszerte több száz programozóból álló szervezete teljesen ingyen és a GNU felhasználási szerződés hatálya alatt terjeszti a K Desktop Enviroment alkalmazást. A KDE célkitűzése, hogy olyan felhasználói felületet nyújtson a különböző Linux-rendszerekhez, mint a Windows



vagy a MacOS (az összes alkalmazás egységes grafikus felülettel rendelkezik) – megőrizve a Unix-rendszerek erejét és megbízhatóságát. A KDE a grafikus felület elemein túl rengeteg alkalmazást tartalmaz. Ezek között hálózati, irodai és multimédiás programok szerepelnek. A rendszerfelügyelet terén is élen jár, mivel számos, a Linux beállítására alkalmas segédprogramot tartalmaz. A teljes felületet magyarították, így használata az angol nyelvet nem ismerőknek sem okozhat gondot. A változatszám-ugrást az az alapvető változás tette szükségessé, mely szerint a jelenlegi változat már a QT 3-as könyvtárat használja. Ez nem csereszabatos a QT 2-vel, azonban az alkalmazások a régi változat megtartása mellett továbbra is futtathatók maradnak. A Troll Tech által fejlesztett és karbantartott QT az egyik legjobb GUI-könyvtárként van számon tartva. Legfőbb előnye, hogy kereskedelmi színvonalú és jól karbantartott könyvtár. Felhasználásának feltételei között azonban olyan megkötések is szerepelnek, mint a termék teljes ingyenessége, a forráskód kiadása, és hogy a QT-könyvtáron semmilyen változtatás nem hajtható végre. Természetesen kereskedelmi jellegű alkalmazások készítése esetén a felhasználási engedély a Troll Techtől megvásárolható. A KDE alkalmazásai közül jelentős változáson ment keresztül az ***X-Terminal***, a ***Konqueror***, a ***Kate***, a ***Kpilot*** és a ***Korganizer***. Különösen sokat fejlődött a ***Konqueror***, az ötödik nemzedékbeli webböngésző, mely az ***Internet Explorer***, a ***Netscape Communicator*** és a ***Windows Explorer*** szolgáltatásait és tudását ötvözi. Fejlődött a böngésző JavaScript-, DHTML- és DOM-megvalósítása, valamint elődeinél érzékelhetően gyorsabb és megbízhatóbb lett. A KDE 3.0 egészen új jellegű csoma-gokat, oktatási programokat is tartalmaz. Ezen túl a francia és a latin nyelvet is gyakorolhatjuk, gépírást tanulhatunk, betekinthetünk a mini planetáriumba, és egy gyerekeknek szánt olvasás oktatóprogram segítségével tökéletesíthetjük csemetéink olvasástudását. Aki még nem próbálta, annak ajánlanám a magyar ***Ispell*** helyesírás-ellenőrző csomagok telepítését, ugyanis a KDE alkalmazásai – a ***Kate***, a ***Kmail*** és a ***Kword*** – is önműködően használják. A legfrissebb magyar Ispell nyelvi csomagok a CD Iroda/Ispell könyvtárban találhatók. A jelenleg elterjedt Linux-változatok mindegyike tartalmazza a futtatásához szükséges ***Ispell*** csomagokat, így használatához csak a magyar helyesírási modulokat kell telepíteni. A rendszerünknek kiadott `ispell -v` paranccsal tájékozódhatunk a program meglétéről és változatszámáról. Ha ez megvan, a `magyarispell-0.83.tar.gz` csomag kibontása következhet, majd fordítása a `make all` és telepítése a `make install` parancs segítségével. A frissen kiadott KDE 3.0 telepíthető forráskódból, vagy az előre fordított Conectiva, Mandrake, Slackware, Gentoo, SuSE és Tr64 bináris változatok segítségével. CD-mellékletünkre a KDE 3.0.1, a Mandrake 8.1/8.2 és a SuSE 7.2/7.3/8.0 Linuxra fordított csomagok első javított kiadása került fel.

Links

A Linux- vagy Unix-rendszert használók talán már találkoztak a kicsi, konzolos felületű, de annál hatékonyabban használható Links böngészővel. Ezentúl a konzolos böngésző jelző nem állja meg a helyét, ugyanis már képes X-környezetben akár FrameBuffer, akár `svgalib` segítségével működni. Helyesen megjeleníti a táblázatokkal, keretekkel és képekkel tarkított oldalakat, rendkívül jól beállítható és grafikus módban is szemtelenül gyors. Természetesen továbbra is használható szöveges módban bármely terminálkörnyezetben. A böngésző 25 nyelven – többek között magyarul is – társalog, és HTML 4.0- (CSS nélkülözve), HTTP1.1- és a JavaScript-támogatással rendelkezik. A támogatott operációs rendszerek között a Linux, a Unix, a BSD,

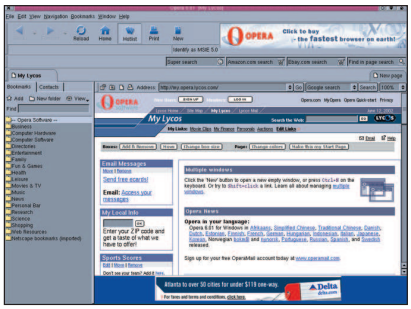
az OS/2, a Cygwin Windows alatt, az AtheOS, a BeOS és a FreeMint szerepel.

MozillaRC3

Megjelent a Mozilla1.0 webböngésző harmadik javított kiadásra jelölt változata. Az 1.0 RC1-hez képest százszámra javították a leggyakoribb leállást vagy lefagyást okozó hibákat. A legfrissebb kiadás további frissítéseket és biztonsági hibajavításokat is tartalmaz, ezért ajánlott az azonnali frissítés. Az RC2-es változathoz képest főleg a lefagyásokat okozó gondok lettek orvosolva. A hibajavítások ellenére az az érdekes helyzet állt elő, hogy a fejlesztők a hibaadatbázis növekedésével nem képesek lépést tartani. Míg az RC1-es változatban 533 gyakori leállást okozó hiba lett jegyezve, az RC2-es változatban ez a szám már 561-re, az RC3-as kiadásban pedig már 585 bejegyzésre növekedett. Természetesen a felmerült hibák a felhasználók nagy részét nem érintik, hiszen a telepítőkészletből mindenki csak a szükséges elemeket telepíti.

Opera

Megjelent Opera Software által fejlesztett 6.0 Linux végleges változata. Az egyre nagyobb népszerűségnek örvendő böngésző főbb előnyei közt szerepel elenyészően kicsi mérete a Netscape és



a Mozilla böngészőkhöz képest, hiszen önmagában alig 3 MB-ot tesz ki. Igaz ugyan, hogy ez a csomag nem teljes, nem tartalmazza a Java- és a levelezéshez szükséges környezetet, így arról mindenkinek saját magának kell gondoskodnia. A legfontosabb tulajdonsága talán az, hogy egyáltalán nem hajlamos az összeomlásra, sőt, a felhasználók által is az egyik legüzembiztosabb böngészőként van számon tartva. A szörfözők kedvenc tulajdonsága a gyors programindulás és a rendkívül fürge oldalmegjelenítés. Az ingyenes változat használatáért cserébe csak egy kis reklámot kell elviselni a kezelőfelületen, ami a böngésző megvásárlása esetén és a regisztrációs szám megadásával természetesen eltüntethető. A jelenlegi kiadáshoz fűzött közlemény szerint az új böngésző elődeinél gyorsabban jeleníti meg a weboldalakat, és ez első olyan linuxos Opera-változat, amely teljes mértékben támogatja a nem latin ábécét használó ázsiai nyelveket. A végleges változat rengeteg hibajavítást, a leggyakoribb leállásokat okozó hibák javítását tartalmazza. Sokat javult a JavaScript-, a CSS-, a Cookie- és a Bookmark-kezelés, valamint az előző változatokkal ellentétben a végleges változatban már működik az önműködő proxybeállítás (automatic proxy configuration). CD-mellékletünkre már az első javított kiadás került, amelyben megszüntették a feltöltés során jelentkező, megváltoztatható fájltulajdonságok biztonsági hibáját.

Netscape

Beindult a fejlesztői gépezet a Netscape háza táján: kiadták a 6.2.3-as változatot, amely nem újdonságokat, hanem többnyire hibajavításokat tartalmaz. A Netscape legfrissebb üzembiztos változata továbbra is a nyílt forrású Mozilla 0.9.4-re épül. Ezzel majdnem egy időben megjelent az első 7-es RP1-változata, mely a Mozilla 1.0RC2-re épül és tartalmazza a Mozillától már megszokott szolgáltatásokat, a többablakos böngészést, amely alkalmas az összenyitott oldal könyvjelzőként történő mentésére; a letöltéskezelőt, amellyel a letöltések bármikor szüneteltethetők, illetve folytathatók; a Favicon.ico megjelenítését a webcímében és a füleken; valamint a nyomtatási előképet, amely a beállítás terén felveszi a versenyt a többi böngészővel.

MPlayer

Teljes gőzzel folyik a magyar fejlesztésű videolejátszó, az MPlayer fejlesztése, mely 0.6-os változatától már a világ legnépszerűbb filmlejátszójaként van számon tartva. Az MPlayer kezeli az MPEG, az AVI, a VOB, az ASF/ WMV, a VIVO, és a QT/MOV fájlokat. Képes a VideoCD, az SVCD, a DVD, a 3ivx és a DivX formátumú filmek lejátszására. A kimeneti meghajtók támogatása teljes: az X11-, az Xv-, a DGA-, az OpenGL-, az SVGAlib-, az fbdev-, az AAlib- és az SDL-meghajtókkal is használhatjuk. Ezenkívül működik néhány kártya alacsonyszintű egyedi meghajtójával, ilyen a Matrox, a 3DFX és a Radeon. A folyamatos fejlesztéseknek köszönhetően hetente, illetve naponta jönnek ki az új változatok. Az utolsó nyilvános kipróbálásra kiadott változat az MPlayer 0.9pre4, amelyben számos hibát kijavítottak. Különösen nagy hangsúlyt fektettek az x11-módban történő teljes képernyős lejátszásra, és a kimenetet ezentúl animált gif-formátumban is menthetjük. Használatához továbbra is a legfrissebb XFree, 2.4-es rendszermag, míg fordításához GCC2.95/3.1 ajánlott.

GCC3.1

Megjelent a GCC 3.1-es változata. A jelenlegi kiadás készítése során nem az új lehetőségek beépítése volt a cél, inkább a minőségre és a meglévő hibák kijavítására törekedtek. A 3.1-es változat sokkal szigorúbban, a szabványoknak megfelelően kezeli a C/C++- és a Java-kódokat, újabb lehetőségként pedig az Ada nyelven írt kódok fordítására is képes. A legfrissebb változat elérte azt az üzembiztos állapotot, amikor már érdemes kényes programokat, rendszermagot vagy MPlayert fordítani.

OpenOffice.org 1.0.0

Teljes gőzzel folyik a linuxos csomagok magyarítása, így pár nappal az OpenOffice.org 1.0.0 irodai csomagjának kiadása után elkészült a honosított változata. Így a felhasználók birtokba vehetik a magyar párbeszédablakokkal és menükkel tarkított, minden igényt kielégítő irodai alkalmazást. Az OpenOffice irodai alkalmazás részét képezi a szövegszerkesztő, a táblázatkezelő, a bemutatókészítő, a rajzoló és az egyenletszerkesztő. Nagymértékben együttműködik az MS Office-szal, így képes a legtöbb manapság elterjedt dokumentumtípus kezelésére, mint a Microsoft Word 95/97/2000, az Excel 5.0/95/97/2000 és a PowerPoint 97/2000. Az irodai csomag GPL felhasználási szerződéssel rendelkezik, így mind az otthoni, mind az irodai felhasználók tetszőleges számú számítógépen és teljesen ingyenesen használhatják. A program nagy előnye, hogy futtatható a Linux, a Solaris és a Microsoft Windows 9x/NT/2000 operációs rendszereken. A nyílt forrású irodai alkalmazás honosított változata magában foglalja a magyar helyesírás-ellenőrző szótárt, a Myspell hu_HU 0.8-as változatát. CD-mellékletünkön az Iroda könyvtárban található meg az időközben kiadott újabb, javított helyesírás-ellenőrző szótár a `hu_HU0.83.zip` és az `OO-042.tar.gz` csomagok formájában.

Rendszermag

A CD Rendszermag könyvtárában található meg a legfrissebb fejlesztői mag, a 2.5.16-os. Jelentős változások történtek az USB- és az IDE-meghajtók terén. A jelenlegi kiadás újdonságaként mutatkozik be a leegyszerűsített changelog, mely ezentúl halandó ember számára is fogyasztható formában kerül a felhasználók elé. Természetesen megmaradt a fejlesztőknek szánt kellően részletes és aprólékos bitkeeper-féle changelog is. Tovább folyik azonban a már egyre kevésbé használt 2.2-es rendszermag építése. A 2.2-es sorozat legfrissebb kiadása a 2.2.21-es, mely számos javítást és frissítést is tartalmaz.

Dankaházi István
(danka@linuxvilag.hu) a Linuxvilág munkatársa. Szabadidejében szívesen úszik és kerékpározik.

Matrox Parhelia

A Matrox régóta nem hozakodott elő új grafikus lapkával. A sokat emlegetett, a rajongók által várva várt G800 nem érkezett meg, helyette viszont kifejlesztették az első 512 bites GPU-t. A Parhelia-512 256 MB DDR memória kezelésére képes, akár 20 GB/s sebességgel. Támogatja az AGP 8× felületet, a nyolcvanmillió tranzisztorból álló lapka 0,15 mikronos eljárással készül, két 400 MHz órajelű RAMDAC-ot

tartalmaz, beépített TV-kódolóval rendelkezik, és DirectX 8.1 támogatást tesz lehetővé.
A Matrox volt az, aki annak idején a DualHead-megoldással megismertette a nagyvilággal a kétképernyős megjelenítést. A cég erre is rá tudott ígérni, TripleHead-megoldásával három kijelzőt is képes kezelni, amelyek segítségével szokatlanul nagy munkafelületet teremthetünk, illetve a játékokat is teljesen újszerű formában élvezhetjük. A cég honlapján egyelőre nem találni az új lapkára épülő kártyát, de valószínűleg már nem sokáig kell várakozni – gondot csak a vételár előteremtése okozhat.
➲ http://www.matrox.com

Lopásvédelem minden szinten

A Xerox egyes eszközeihez amolyan elektronikus iratsemmisítőt is ajánl. A komolyabb nyomtatókban, digitális fénymásolókban saját merevlemez található, amelyen a készülék átmenetileg vagy huzamosabb ideig tárolhatja a kinyomtatott iratokat. Ezek – akárcsak a számítógépek merevlemezén a törölt állományok – egészen addig fennmaradhatnak, amíg egy másik dokumentum bitjeivel felül nem írjuk őket. Ha bizalmas anyagról van szó, ez biztonsági szempontból aggályos lehet, így a cég lehetővé teszi a kérdéses lemezterületek felülírását. A szolgáltatást a Document Centre 460, 470, 480 és 490 készülékekhez külön kiegészítésként már most is meg lehet rendelni, az újként bevezetett termékeken pedig alapszolgáltatásként lesz elérhető.
➲ http://www.xerox.com

Vezetéknélküli pénz?

A Visa International és az SK Telecom megegyezése szerint a két cég intelligens kártyákon és infravörös átvitelen alapuló fizetési rendszert épít ki. A rendszert két fontos szabványra építkezve hozzák létre: egyrészt az EMV (Europay-MasterCard-Visa) általános intelligens kártyaszabványra, másrészt az IrFM-re, amely pénzügyi műveletek vezeték nélküli továbbítását teszi lehetővé. A rendszer segítségével a Visa kártyák tulajdonosai – amennyiben az SKT előfizetői – egyszerűen, infravörös átvitel segítségével fizethetnek termékekért. Adataikat mobiltelefonjuk segítségével továbbíthatják majd az értékesítési pontokon elhelyezett leolvasókhoz, természetesen biztonságos módon.
A rendszert egyelőre húsz kereskedő közreműködésével próbálják ki, üzembe helyezése után azonban mintegy 30 ezer üzlet és kétmillió koreai előfizető részvételére számítanak.
➲ http://www.visa.com

Új ügyvezető az Oracle Hungary élén

2002. június 1-jétől ***Füzes Péter*** tölti be az Oracle Hungary Kft. ügyvezető igazgatói posztját. Füzes 36 éves mérnök-közgazdász, diplomáit Budapesten szerezte. Karrierjét a Novotradenél kezdte, 1991-től az Internet Kft.-nél dolgozott, amelyet 1994-ben megvásárolt az AT&T. Az új cégben az értékesítéselőkészítést és ügyfélkezelést végző csapat vezetője lett. 1996-tól a Lucent Technologies Magyarország Kft. ügyvezető igazgatójaként folytatta pályáját, később a Lucent Technologies Inc. regioónális igazgatójává is kinevezték. 1996-tól a Lucent cég szétválása után az üzleti kommunikációs rendszerek üzletágat folytató jogutód, az Avaya Magyarország Kft. közép-európai regionális ügyvezető igazgatói tisztségét látta el.
Füzes Péter ***Stefan Ström***-öt követi a poszton, aki az Oracle Észak- és Közép-Európáért felelős elnökhelyettesi tisztjének betöltése mellett irányítja az Oracle Hungary-t 2001. november 29. óta.

MMS-tanácskozás Budapesten

2002. április 18-án az Ericsson csúcstechnológiát képviselő rendszerén a világon elsőként a Westel indította el a teljes körű kereskedelmi multimédiás üzenetközvetítési szolgáltatást (MMS).
Az MMS világpremierjét követően a nagy érdeklődésre és a rendkívül kedvező nemzetközi fogadtatásra tekintettel az Ericsson Magyarország és a Westel Mobil Távközlési Rt. úgy döntött, hogy nemzetközi tanácskozást rendez Budapesten, amelyen bemutatja az immár másfél hónapja működő szolgáltatást, illetve ismerteti az eddig elért eredményeket és tapasztalatokat. A június 4-én megtartott fórumon az előadók a Westel felkészüléssel, szolgáltatásindulással kapcsolatos tapasztalatairól, a technikai kérdésekről, a piaci várakozásokról és a működő tartalomszolgáltatásokról nyújtottak átfogó képet. Nem csupán magáról az MMS-rendszerről, hanem az Ericsson Consumer Lab szervezetén belül folyó, a felhasználói szokásokat vizsgáló kutatásokról is szó esett. Az MMS sikerének elengedhetetlen feltétele a sokszínű és megbízható tartalom, ezért a Westel már a szolgáltatás beindításának első napján útjára indította MMS információs szolgáltatásait. A tanácskozáson bemutatót tartott az Ericsson partnerei közül több tartalomszolgáltató, köztük két hazai cég is, felvillantva néhány lehetséges alkalmazási területet.

40 hüvelykes LCD-TFT tévé a Samsungtól

A Samsung bemutatta a világ első 40 hüvelykes TFT-LCD kijelzőjét. A kijelzőt egy különleges megoldással kimondottan tévéadások megjelenítésére fejlesztették. Képpontjainak válaszideje 12 ms, így mozgókép visszaadására tökéletesen alkalmas, láthatósági szöge 170 fok, képaránya 15:9. A Samsung – és vele együtt más gyártók – már eddig is kínáltak kisebb-nagyobb TFT-LCD tévéket, ám hasonló méretű kijelzőket műszaki akadályok miatt tömegesen még nem gyártottak. A Samsung tavaly augusztusban készítette el szokatlanul nagy kijelzőjének első próbapéldányát, a sorozat gyártásra egészen eddig várni kellett. A készülék tényleges kereskedelmi elérhetősége és ára egyelőre ismeretlen.
➲ http://www.samsung.hu

Nagyteljesítményű Fujitsu merevlemezek

A hazánkban kevésbé kedvelt – tegyük hozzá, ok nélkül, hiszen a cég a második legnagyobb merevlemez-szállító volt ez év első negyedében a vállalati piacon – Fujitsu merevlemezeinek családja újabb nagyteljesítményű sorozattal bővül. Az új MAP-sorozat tagjai 10, a MAS jelzésű példányok pedig 15 ezres fordulaton működő lemezeket tartalmaznak.

Természetesen nemcsak a lemezek fordulatszáma nőtt, hanem belső átviteli sebességük, valamint kapacitásuk is, amely elérheti a 147 GB-ot. Mindkét típushoz 8 MB 32 bites gyorsítótár tartozik, a rendszerhez Ultra320 SCSI vagy FC-AL felületen keresztül csatlakozhatnak. A MAP-sorozat már júliustól elérhető lesz, a MAS-sorozatra viszont szeptemberig várni kell.
➲ http://www.fujitsu.com

Tőkeemelés az Axelerónál

A Matáv úgy határozott, hogy a további sikeres növekedés elősegítése érdekében négymilliárd forinttal megemeli a társaság saját tőkéjét. 2001 végén az Axelero saját tőkéje 2,5 milliárd forintot tett ki. A Matáv arról is határozott, hogy ***Drajkó László***t, az Axelero vezérigazgatóját és ***Simó György***öt, a társaság általános vezérigazgató-helyettesét, valamint ***Dr. Nagy Bálint***ot, a Matáv csoport kommunikációs igazgatóját az Axelero igazgatósági tagjának nevezi ki.

Szuperbiztonságos Linux

A fejlesztők sikeresen fordították le a Security-Enhanced Linux (SELinux) egyik próbaváltozatát – igaz, egyelőre csak modulként.

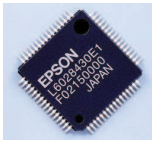
Az amerikai NSA segédletével készülő Linux az eredeti szándék szerint nem lesz teljes teljesztés, ám mindenképpen fontos hatással lesz majd az összes Linux-változatra – állítja ***Grant Wagner***, az NSA egyik vezető tisztviselője.
A fejlesztések az operációs rendszer még biztonságosabbá tételét célozzák, többek közt szolgáltatásait a meglévő önkényes mellett meghatalmazás jellegű hozzáférés-védelemmel is bővíteni szeretnék. Természetesen az NSA sem jótékonysági alapon dolgozik együtt egy nyílt forrású program híveivel; a biztonságosabb rendszereket az amerikai kormányhivatalokban is használni fogják.
➲ http://www.nsa.gov/selinux/index.html

Elkészült a Palm OS 5

A Palm bejelentette, hogy a fejlesztők számára már elérhető a Palm OS 5-ös változata. Az új zseb-operációs rendszer alá már nem a megszokott Motorola Dragonball processzorok, hanem nagyobb teljesítményű ARM-lapkák kerülnek. A nagyobb erőre szükségük is van az új Palm-gépeknek, hiszen csak így vehetik fel a versenyt a Microsoft Windows-alapú multimédiás készülékekkel. A Palm nagy hangsúlyt helyez az áttérés – várhatóan hosszadalmas – időszakának zökkenőmentességére, így ígéretük szerint a régi programok körülbelül nyolcvan százaléka az új alapokon is működni fog.
➲ http://www.palmos.com

Instant USB

A Seiko Epson elkészítette az első USB On-The-Go vezérlőlapkát. A jelenlegi USB-csatlakozós eszközöknél a kapcsolat egyik résztvevője mindig a személyi számítógép. Hiába van két USB-s eszközünk, közvetlen kapcsolatot nem tudunk teremteni közöttük – ezt oldja meg az USB On-The-Go.



Elsődleges célterület a digitális fényképezőgépek és nyomtatók piaca, hiszen az új megoldással közvetlenül a fényképezőgépről is átküldhetjük a nyomtatónak a képeket, nem lesz szükség számítógépre.
➲ http://www.epson.co.jp

Enged a Microsoft

A Windows apró cafatokra cincált változatának bemutatása ugyan elmaradt, ám a Microsoft a felhasználók számára lehetővé fogja tenni, hogy az elektronikus levelezésre, böngészésre és azonnali üzenetküldésre használt programokat a saját ízlésük szerintire cseréljék. Az alkalmazások nem kerülnek tényleges eltávolításra a rendszerből, ám az mégis úgy fog viselkedni, mintha a törlés megtörtént volna. Az új lehetőség a Windows XP-hez várhatóan augusztusban ingyenesen megjelenő javítócsomaggal válik elérhetővé. A lépésre az éppen folyó antitröszt per révén sikerült a Microsoftot rávenni. A Windows korábbi változatait a cég nem módosítja. Természetesen abban eddig sem akadályozta senki a felhasználókat, hogy saját böngészőt vagy egyéb internetes programokat telepítsenek, de a beépített alkalmazások eltüntetését a rendszer nem támogatta. Még fontosabb, hogy az OEM-gyártók előtt megnyílik annak lehetősége, hogy már az operációs rendszer előtelepítésekor módosítsanak bizonyos összetevőket, így például Netscape vagy Opera böngészővel „szereljék fel” a gépet.

Handspring Treo 90 billentyűzettel

A Handspring új, Treo 90-es készüléke friss tagként belépett a billentyűzettel is rendelkező zsebtitkárok klubjába. Az apró billentyűzet a megszokott gombok fölött kapott helyet, feladata az adatbevitel gyorsítása – ennek ellenére a Háború és béke begépelését nem ezen fogjuk elkezdeni. A 300 dolláros Treo egyébként 16 MB memóriával kapható, további bővítését Secure Digital és MultiMediaCard kártyákkal végezhetjük el. Színes kijelzővel és Li-ion akkumulátorral rendelkezik, súlya megközelítőleg 110 gramm.
➲ http://www.handspring.com

CD-nyi CD-író

Az Archos Cesar nevű meghajtója pontosan másfél centiméterrel szélesebb, mint egy 120 milliméteres korong. Kicsiny mérete ellenére remekül elboldogul a lemezekkel, írni és újraírni nyolcszoros, olvasni pedig 24-szeres sebességgel tudja őket. A cég Micro Power Management nevű fejlesztésének köszönhetően külső tápegységre sincs szüksége, közvetlenül a PC-ről vagy Macintoshról is nyerheti az áramot. A készülék USB 2.0, FireWire vagy PC-kártyás felülettel rendelkezhet, súlya 400 gramm, ára pedig 290 euró.
➲ http://www.archos.com

Néhány srác még hiányzik

A Mozilla-tervezet újfajta felhasználói szerződéssel jelenne meg – ám egy apróság hátráltatja a folyamatot. Több mint 300 fejlesztő már hozzájárult a változáshoz, ám vannak néhányan, akik egyszerűen elérhetetlenek. A Mozilla webhelyén külön oldalt létesítettek a hiányzó betyárok felkutatására, ahol a nagyérdemű segítségét kérik. Szerencsére az eltűntek nagy része már előkerült, lapzártakor mindössze négy nyomtalanul felszívódott úriember szerepel a listán.
➲ http://mozilla.org/MPL/missing.html

Mobiltelefonról vezérelhető számítógépek Japánban

Az IBM Japan Desktop On-Call Gateway nevű programjával nem kevesebbre vetemedhetünk, mint hogy távolról, mobiltelefonunk segítségével vezéreljük számítógépünket – feltéve, hogy el tudjuk érni az NTT DoCoMo szolgáltatását, és megfelelő készülékkel is rendelkezünk. A programot például a vállalati demilitarizált zónába kell telepíteni, így válik lehetővé a vállalati hálózaton található PC-k távvezérlése. A kiszolgáló SSL-en vagy a mobiltelefon sorozatszáma alapján azonosítja a hívót, aki ugyan munkaasztalának egyszerre mindig egy kis részét látja, ám a megfelelő számgombokkal tetszése szerint navigálhat a képernyőn, és egérkattintásokat is küldhet munkaállomásának.

Megjelent a Mozilla 1.0

Hosszas fejlesztés után megjelent a Mozilla 1.0-s változata. A nem csak asztali

gépekre elérhető Gecko-motorra épülő böngésző, valamint a köré csoportosuló alkalmazásegyüttes nyílt forrású fejlesztés eredménye. A program támogatja az ismert webes szabványokat, így többek közt a HTML 4.0-t, az XML-t, CSS1 és CSS2t, az Unicode karaktereket és a JavaScript 1.5-ös változatát. A fejlesztés során nagy hangsúlyt helyeztek a könnyű honosíthatóságra, talán ennek is köszönhető, hogy a Magyar Mozilla Projekt munkája révén a Mac OS, Linux, AIX, BSD, Windows és OS/2 operációs rendszerek alatt futó program magyarul is elérhető.
➲ http://mozilla.rog

Solaris 9

Az operációs rendszer kifejezés új jelentéssel töltődik meg – valami ilyesmi történik a Sun szerint a Solaris 9 megjelenésével. Az új rendszer számos olyan új vagy javított szolgáltatással bír, amelyek egyfelől a biztonságot, másfelől a könnyebb felügyelhetőséget segítik elő. Különösen ez utóbbi fontosságát hangsúlyozzák a fejlesztők, hiszen minél kevesebb felügyeleti munka akad egy-egy kiszolgálóval, annál könnyebben csökkenthetők a birtoklás és üzemben-tartás költségei. Néhány az új fejlesztések közül, csak címszavakban: IPSec Internet Key Exchange-támogatással, véletlenszám-generátor, biztonságos LDAP és teljes körű Smart Card-támogatás, számos új felügyeleti alkalmazás, Live Upgrade 2.0. Az új Solaris alapváltozata akár ingyenesen is beszerezhető, mindössze a CD-t és a kezelési költséget kell megfizetnünk.
➲ http://www.sun.com

IBM-programok oktatási intézményeknek

Az IBM Magyarországon is meghirdette nemzetközi felsőoktatási programját, melynek keretében számos programját egyetemek, főiskolák számára ingyenesen hozzáférhetővé teszi, közöttük a WebSphere e-business programfelületet, a DB2 adatbázis-kezelőt, a Lotus irodai alkalmazásokat és több fejlesztőeszközt. Az érdeklődő intézmények a
➲ http://www.ibm.com/university/scholarsprogram címen jegyezhetik be magukat.
Az IBM Magyarországi Kft. elsőként a Budapesti Közgazdasági és Államigazgatási Egyetemmel kötött megállapodást, melynek alapján az intézmény hallgatói és oktatói az IBM különféle programjait – nem kereskedelmi célokra – ingyenesen használhatják. A felkínált alkalmazások az intézmény számítógépein szabadon futtathatók, és korlátozás nélkül használhatók fel oktatási, kutatási és tanulási célokra. A cég az ilyen módon átadott programokhoz rendszeres frissítést és állandó elérésű támogatást biztosít. Az ingyenesen elérhetővé tett alkalmazások értéke kereskedelmi forgalomban megvásárolva meghaladná a több százezer forintot. A programokon kívül az IBM térítésmentesen bocsátja rendelkezésre oktatási és képzési anyagait, számos leírást, tanulmányt és elektronikus formában elérhető tankönyvet is.

Medgyesi Zoltán (mzx@axelero.hu) a BMGE 24 éves informatika szakos hallgatója. Szabadidejét legszívesebben a barátnőjével tölti. Szeret autózni és bográcsban főzni. A Linuxot hat éve ismeri, de még nem volt lelkiereje, hogy áttérjen rá. A Linuxvilág magazin hírszerkesztője.

Kiszolgáljuk!

Az, hogy az új kiszolgálóra Linux kerüljön-e, a legtöbb felvilágosult helyen nem lehet kérdés – hacsak nincs valamilyen különleges követelmény, ami miatt egy másik operációs rendszert kell választani. De milyen vas kerüljön alá? Itt kezdődik a hajhullást és álmatlan éjszakákat eredményező beható fejvakarás.

Ahhoz nem kell tehetség, hogy elmenjünk a boltba, megvegyük mindenből a legnagyobbat, a legdrágábbat, no meg amelynek a neve mellé odaírták, hogy „server". Anyagi okokból általában erre nincs is lehetőség. Marad tehát a gondos tájékozódás, majd a válogatás, végül az alkatrészek és egy szatyorravaló hűtőventilátor megvétele.

Egy másik megoldás az, hogy vásárolunk egy márkás gépet. Nem biztos, hogy jobbat fogunk kapni, mintha minden tudásunkat beleadva sajátot állítanánk össze, ám reménykedhetünk abban, hogy egymással jól együttműködő alkatrészekből állították össze, és olyan garanciát és szolgáltatásokat, valamint kiegészítőket kapunk hozzá, amik miatt érdemes mellette dönteni. A HP is rendelkezik olyan kiszolgálónak összeállított gépekkel, amelyek gyakorlatilag bármelyik felhasználó asztalán előfordulhatnak. A legújabb ilyen a tc sorozat, melyből a tc2100-as modell járt nálam.

A gépet körülvevő hatalmas dobozról nem írnék tanulmányt. A kellékek halmaza viszonylag kis elemszámúnak mondható, és a géphez tápkábelt, egyszerű billentyűzetet és mezei egeret kapunk, valamint két CD-t, amelyekről később lesz szó.

Maga a gép külcsínét tekintve nekem nagyon tetszett. A tervezők felrúgták az unalmas fehér színű fémlemezből összeállított házak nyomasztó hagyományát, és a gép zsigereit tetszetős fekete műanyagházba burkolták. Természetesen a ház – amelynek homlokzatán formatervezett HP felirat díszeleg – nem tisztán műanyagból készült, az elektromágneses zavaró jelek ki- és bejutását egy belső fémborítás, az illesztéseknél pedig vékony fémlamellák akadályozzák meg. A ház természetesen lezárható, a doboz hátulján egy fekete műanyag bumpszli éktelenkedik, ez a lakat rajta. A doboz oldala nem csavarozható, hanem egy kallantyúval nyitható, bezárt állapotban ennek mozgatását akadályozza meg egy fémpöcök.

A gép belseje roppant célszerűnek tűnt. Legfontosabb előnyének azt tartottam, hogy a merevlemezek nem a ház hosszában, hanem keresztben helyezkedtek el benne. Ennek köszönhetően ha bővíteni, szerelni szeretnénk, esetleg az átkötéseket kívánnánk átrendezgetni a meghajtók hátulján, nem kell őket kihúznunk a helyükről – letarolva ezzel a gép teljes belsejét –, hanem könnyedén és fájdalommentesen, csupán a gép oldalát leemelve tényleges kedhetünk.

A processzorra termetes hűtőborda került, érdekessége, hogy ventilátora nem a processzor felé, hanem azzal párhuzamosan fújja a levegőt. Meggyőződésem, hogy ezzel elkerülhető a por lerakódása a borda processzor felőli részén, és nem kell tartani a hűtés hatékonyságának leromlásától.

A gép belsejéről általában is elmondható, hogy igényesen szerelt. A kábelek a helyükön vannak, szépen, szögletesen, katonás rendben sorakozik belül minden. Mivel gyárilag csak egy merevlemez van a dobozban, alatta táp- és adatkábelcsatlakozás és üres bővítőhely – elkülönített csavarokkal együtt – várja a második meghajtót. Már épp ki akartam osztani egy pirospontot az igényes belsőért, amikor megdöbbenten vettem észre, hogy az egyik USB-csatlakozóra rálóg a hátoldali fémlap egy apró darabja – alighanem félrecsúszott valami egy kicsit az összeállításnál. Némi izomerővel és egy csavarhúzóval lehet rajta segíteni, csakhogy nem azért vásárol az emberfia márkás gépet, hogy hanyagul legyen összeszerelve.

A meghajtók mindegyike sínekre szerelhető, majd egyetlen mozdulattal a helyére csúsztatható. Merevlemez összesen kettő kerülhet a házba, a gyári CD-meghajtó alatt további két nagy meghajtó fér el, felülről pedig a hajlékonylemezes meghajtó figyeli fenségesen társait – lehet, hogy a rendszerindításra alkalmas CD-k korában én a hagyományokkal szakítva kihagytam volna a gépből, ez azonban csak magánvélemény.

A gép – elsősorban a processzor – megfelelő hűtésébe egy pótventilátor is besegít. A tc2100-as még vele együtt is meglepően csendes, működése közben nyugodtan alhatunk mellette.

A tc2100 kizárólag minőségi alkatrészeket tartalmaz, ennek megfelelően nem is sikerült zavarba hoznom. Az órákig, napokig tartó folyamatos másolgatás és tartós üzemelés meg sem kottyant neki, semmilyen hibát vagy lefagyást nem mutatott – itt jegyzem meg, hogy az UHU Linux első próbaváltozata is remekül teljesített.

A gép felépítése:

- közepes méretű toronyház két merevlemezhellyel, három 5,25"-es hellyel és egy hajlékonylemezes meghajtónak kialakított hellyel, továbbá 250 W teljesítményű Lite-On táppal;
- Asus P4B-MX alaplap beépített hálózati csatolóval, egy AGP-, három PCI- és három SDRAM-foglalattal;
- nVidia Riva TNT2 lapkás, AGP-foglalatba illeszkedő VGA-kártya;
- Maxtor D740X-6L típusú, 40 GB kapacitású, 7200-as fordulaton működő merevlemez;
- Lite-On 48×CD-ROM;
- 1,7 GHz órajelű Pentium IV processzor;
- 128 MB Micron Technologies PC133-mas, ECC Registered memória.

Megbízható gépet azért építünk, mert minél nagyobb rendelkezésre állást szeretnénk elérni. Meghibásodásokra azonban mindig számítani kell, és talán kijelenthetjük, hogy ez önmagában még nem tragédia. Sokkal inkább baj, ha a javítás, esetleg a biztonsági mentések

visszaállítása órákig, netalán napokig tartó leállást, a szolgáltatás kiesését eredményezi. A HP több olyan garanciakiterjesztést is kínál gépeihez, amelyek révén a kiesés időtartama jelentősen csökkenthető, ezek közül – tájékozódásom eredménye szerint – hazánkban kettőt vásárolhatunk meg.
Az egyik lehetőség, hogy a HP karbantartója felárért négy órán belül megjelenik, és megkezdi a gép javítását, feltéve, hogy munkanap van, és még beleférünk a munkaidőbe. Előzetesen telefonon kérdeznek ki minket a hibajelenségről, így a megfelelő alkatrész cseréje azonnal megtörténhet, és a kiszolgáló jó eséllyel újra működhet. A szolgáltatásért körülbelül 110 ezer forintot kérnek áfával együtt.
A másik ajánlat másnapi kiszállást ígér, ám ha jól értettem, nem csak munkanapokon. Erre utal magasabb, 160 ezer forint feletti ára. Mindkét szolgáltatás három éven keresztül vehető igénybe.
Első, sőt, még második hallásra is magasnak tűnnek ezek az árak, ám nem szabad elfeledkezni arról, hogy nemcsak a gép javításáról van szó, hanem az alkatrészek cseréjéről is. A szerelés egy rendszergazdának aligha okozna gondot, ám adott esetben az alkatrész beszerzése annál inkább. És még egy fontos szempont: ha bármilyen gond akad, másra tudunk mutogatni.
A Toptools a HP általános eszközfelügyeleti programja. Segítségével az asztali gépek hőmérsékletétől kezdve egészen a nyomtatók állapotáig rengeteg dolgot felügyelhetünk – akár egészen nagy hálózatokban is. Az eseményeket és állapotadatokat ügynökök segítségével gyűjti be a hálózatról – ilyen ügynököket azonban sajnos csak a különféle Windows-változatok alá kínálnak. Előnye viszont, hogy képes együttműködni a nagy felügyeleti rendszerekkel, így a HP OpenView Network Node Manager, a Microsoft SMS, a CA Unicenter TNG, az IBM Tivoli Enterprise Management és Tivoli NetView termékeivel.
A CD-k tartalma:

az első lemezen
HP Toptools Device Manager
HP Web Jetadmin
Device Agents

a második lemezen
HP Toptools for NNM
(NT, Unix, Solaris)
HP Toptools for Microsoft SMS
HP Toptools fo Unicenter TNG

Adatok

E-mail: ➲ http://www.hp.hu/vevoszolgalat/vevo.asp
➲ http://www.hp.hu

Medgyesi Zoltán (mzx@axelero.hu)
a BMGE 24 éves informatika szakos hallgatója. Szabadidejét legszívesebben a barátnőjével tölti. Szeret autózni és bográcsban főzni. A Linuxot hat éve ismeri, de még nem volt lelkiereje, hogy áttérjen rá. A Linuxvilág magazin hírszerkesztője.

Vajon hány haiku születik még, mire ráununk?

E heti jelöltünk „A hét szabad programja” címre a ➲ http://www.oblomovka.com/code/haiku/haiku. Ez egy olyan Python-parancsfájl, amely „véletlen” haikukat, azaz 17 szótagos mondatokat keres bármilyen szövegfájlban, ahogy mi is tettük az LDP HOWTO (HOGYAN) leírásokkal még valamikor 2000-ben. Egy Linux levelezési lista a következő haikukat tartalmazta (figyelem: a fordítások már nem 17 szótagból állnak):

- Nos, a Junkbusterrel nem tudom megnyitni, de minden bizonnyal csodálatos.
- Inkább lerágom a lábam, minthogy még egyszer Netscape webkiszolgálót használjak.
- Solarisos pelenkások és lükék: a `/opt/foo` fájl helyesen `/usr/lib/foo`, a mindenségit!
- Az Orinoco hatósugara messze a legnagyobb, és nagyon jó vezérlő van hozzá.
- Bush irodalmi művek iránti rajongása igen fiatal korában megmutatkozott.
- Hinned kell nekünk. Nincs más választásod. INTERAKTÍV SZÁMÍTÓGÉPES SZOLGÁLTATÁS.
- Nielsen soha életében nem tervezett semmi szépet.
- A székesegyházban a fenntartott helyek egy része le van nekünk foglalva.

Még *Richard Stallman* is (haiku)költő, csak nem tud róla. Ez a haiku a ➲ http://gnu.org webhelyről származik, és a védjegyes programok által okozott gazdasági kárra vonatkozik:
„Egyvalaki szert tesz egy dollár haszonra, miközben kétdollárnyi értéket pusztít el.”
E havi jelöltünk „A hónap szabad programja” címre a mencal menstruációs naptár, amely nagyon hasonlít a Linux programnaptárához. Mindössze annyi a különbség, hogy a mencalban bizonyos napok piros színnel jelennek meg: ➲ http://mencal.kyberdigi.cz/english.html

Don Marti

Linux Journal 2002. június, 98. szám

UnitedLinux

Szövetséget kötött a SuSE, a Conectiva, a Caldera és a Turbolinux.

Újabb linuxos érdekcsoport született. 2002. május 30-án négy vezető linuxos cég megalapította a UnitedLinuxot, amely szabványalapú, világméretű Linux-megoldásként szeretne bekerülni a történelembe, és egyelőre csak az üzleti felhasználókat célozná meg.
A UnitedLinux fejlesztését négy szövetséges végezi, mely vállalati szintű, ipari szabványokra épülő, megbízható, egységes felületű alkalmazás lesz, és ami a gép- és alkatrészszállítók számára egyedülálló, magas szintű Linux-megoldások kínálatát teszi lehetővé.
Az IDC piaci- és trendkutató vállalat felmérése szerint tavaly az észak-amerikai és nyugat-európai vállalatok negyven százaléka már használta a Linuxot. A UnitedLinux a vállalati környezetben is fel szeretné gyorsítani a Linux befogadását azáltal, hogy az üzleti alkalmazások, valamint a tanúsítvánnyal rendelkező gépek és alkatrészek nagyobb kínálatát kezdeményezi. Véleményük szerint a vásárlók előnyként értékelik majd az átfogó terjesztési stratégiát, ami csak abban az esetben jelenthet kézzelfogható hasznot a linuxos társadalom számára, ha a multinacionális cégek szintjén képes hirdetni. Kínálja a vállalati termékek nemzeti nyelven való elérhetőségét, valamint az oktatást, a terméktámogatást – ami az eddigi tapasztalatok szerint meglehetősen ingoványos területnek bizonyult. Hazánkban például eddig csak egyetlen igazán jól működő segítő rendszer született, amely a ➲ linux-kezdo@mlf.linux.rulez.org címen érhető el.
A UnitedLinux egységes Linux-kóddal szolgál az IBM teljes eServer-terméksorozatára, az AMD 32- és 64-bites, valamint az Intel x86 és Itanium processzorcsaládokra, továbbá támogatja az LSB-t (Linux Standard Base), a Li18nux és a GB 18030 szabványokat. Angol, német, francia, olasz, japán, koreai, portugál, magyar, spanyol, egyszerűsített és hagyományos kínai nyelven telepíthető.
A konzorcium célja, hogy egységes mederbe terelje a Linux-fejlesztéseket. Ez természetesen rendkívül hasznos lehet, hiszen a kezdeményezést az ismertebb gépgyártók támogatják: az AMD, a Borland Software Corporation, a Computer Associates, a Fujitsu Siemens, a Fujitsu Japan, a Hewlett Packard, az IBM, az Intel, a NES, a Progress Software és a SAP. Ez nagyon fontos, hiszen így az eszköztámogatás lényegesen felgyorsulhat. Mint minden jónak tűnő kezdeményezésnek, ennek is lehetnek hátulütői. Eddig azért bíztunk annyira a Linuxban, mert mindig betekinthettünk a forrásokba, a jelek szerint azonban ez a UnitedLinux-felhasználóknak nem feltétlenül adatik meg. Még belegondolni is ijesztő, mi mindent lehet egy program kódjában elrejteni. Jusson eszünkbe az a bizonyos, jól ismert operációs rendszerben futó szövegszerkesztő, amelyben többek között egy „repülőgép-szimulátort" is elrejtettek (húsvéti tojás). Ezzel a felhasználók adatainak biztonsága máris megkérdőjelezhető.
Nézzük meg közelebbről, hogy a fenti együttműködés mit is rejt magában. Lényegében felosztották egymás között a bolygónkat: a Caldera Észak-Amerikában, a Conectiva Dél-Amerikában, a SuSE Európában, a Turbo-Linux Ausztráliában és Ázsiában a legelterjedtebb. Vajon mi történik, ha más is csatlakozni kíván a csapathoz, és ezzel versenytársa lesz a jelenlegi cégek valamelyikének? Hivatalosan nyitottak, tehát bárki csatlakozhat, csakhogy arról mélyen hallgatnak, hogy ez milyen anyagi feltételeket és egyéb garanciavállalást kíván.
Elég meglepő, hogy a Red Hat Linux kimaradt. Vajon miért? Talán mert versenytársa lehetne a Calderának? Esetleg csak azért nem szerepel, mert a Red Hat egyedül is fel tudja mutatni mindazt, amit a UnitedLinux kínál? Hiszen jól használható a terméktámogatásuk, már rég meghódították a „nagyvállalati" piacot, terjesztéseik nyelvi nehézségekkel sem küszködnek, ráadásul a nevük is meglehetősen jól cseng a piacon, de arról sem szabad megfeledkezni, hogy szintén RPM-csomagokat használnak. Még azt is meg merem kockáztatni, hogy a Red Hat tud a legtöbbet az RPM-csomagokról… Tehát nem kizárt, hogy a Red Hat nem kívánt részese lenni ennek a csoportosulásnak, vagy csak nem hívták meg, hogy jelenlétével ne sértse mások érdekeit.
Az eseményeket kívülállóként szemlélve úgy érzem, hogy a UnitedLinux megszületése érdekében szövetséget alkotó cégek célja igazán megnyerő, viszont abban az esetben, ha ismét a Caldera 2000-ben megjelent 3.0-s munkaállomás és kiszolgáló termékében szereplő „per-seat licensing" (munkahelyenkénti felhasználás) megvalósításán fáradoznak, már közel sem olyan vonzó. A Caldera 2000-ben visszavonulót fújt, és ezzel a lépésével egy sereg felhasználóját magára haragította.
Richard M. Stallman, az FSF (Free Software Fondation – Szabad Szoftver Alapítvány) atyja egyenesen arra szólítja fel a linuxos társadalmat, hogy kerüljék a UnitedLinuxot, mivel szerinte nyilvánvaló, hogy terjesztésükben a „perl-seat"-típusú felhasználási szerződést kívánják használni. Időközben a SuSE hivatalosan is kijelentette, ől semmiképpen nem támogatják ezt. Stallman, dörzsölt „öreg" róka, pályafutása során sokszor élt már át olyan helyzeteket, amelyeket mi még a hírből is alig ismerünk, így nem kizárt, hogy Ő látja jól a helyzetet.
Akkor innen viszont már csak egy lépés, hogy egyértelműen lássuk, ennek az érdekcsoportnak csak egyetlen nagy vesztese lehet, a SuSE, hiszen a többiek eddigi tevékenységük folytán az esetleges bukást könnyebben kiheverhetik.

Aki kimarad, az lemarad?

Ezt sajnos nem lehet előre látni. Abban az esetben, ha a UnitedLinux valóban megerősödik, és a vállalati piac után az otthoni felhasználókat is megpróbálja meghódítani, sok száz Linux-terjesztés és linuxos cég kerülhet könnyen a nagy vesztesek közé. Az igazán nagy vesztes ebben

az esetben a Red Hat lehet. Az is elképzelhető, hogy a UnitedLinux nem tudja beváltani a hozzá fűzött reményeket, akkor viszont a konzorcium tagjai olyan presztízsveszteséget könyvelhetnek el, amiből talán csak a SuSE tud komoly anyagi veszteségek nélkül kilábalni.
A felhasználói tömegeket nézve nem feledkezhetünk el a Debian GNU/Linuxról sem. A jelek szerint a UnitedLinux körül kavargó indulatok, érvek és ellenérvek messze elkerülik. Ez talán abból is adódik, hogy a Debian GNU/Linux alapítványi formában működik, így a Linux elüzletiesedésére irányuló kísérletek, vagy az RPM (Red Hat Package Manager) csomagkezelők közötti változatháborúkkal szemben ellenálló – adódik ez abból is, hogy csomagkezelője `dpkg`, azaz *.DEB* kiterjesztésű csomagokat használ. A Debian/GNU Linux biztonságban érezheti magát, mivel felhasználóinak tábora ma már annyira széles, hogy a kritikus tömeget rég meghaladta. Népszerűségét nemcsak legendás megbízhatóságának köszönheti, hanem annak is, hogy a Linux eredeti filozófiájához ez áll a legközelebb. Felmerül a kérdés, hogy akkor most melyik irányt érdemes követni?
Szavazzunk bizalmat a Linux egységesítésére törekvőknek, vagy egyszerűen csak várjuk meg, mit hoz a jövő, esetleg maradjunk a kitaposott stallmani, vagy írhattam volna azt is: Debian GNU/linuxos úton?

Gibizer Tibor
(gibzo@linuxmania.hu)
Újságíró, immár hét éve a Linux elkötelezett híve. Imádja a kutyákat, a kerékpározást és az autós csavargást.

Ők mondták

A TiVo csak egy rossz felület csomagolása. *(Bruce Sterling)*

Nem tetszik, hogy egyre divatosabbá válnak azok a dolgok, amelyek aláásható a Google megbízhatóságát és pontosságát. *(Seth David Schoen)*

Csak remélni tudom, hogy nem fogják „szoftverterroristáknak" nevezni azokat, akik digitális adathordozókat másolnak. *(Derek K. Miller)*

Naiv elképzelés volt azt hinni, hogy a Háló nyitott maradhat. Soha semmi nem marad nyitott. A dolgok mostanában nem alakulnak jól – az üzleti modelleket védő, magas szintű vállalati érdekcsoportok újult erővel lobbiznak azért, hogy káros és ostoba törvényeket fogadtassanak el. *(Mitch Kapor)*

Hollings törvényjavaslata bizonyos értelemben visszaesés. Fenyegeti mindazt, amit az emberek el szeretnének érni, bármely oldalon álljanak is… Az iparban dolgozók nem akarnak olyan szorosan összetartozó forgatókönyveket, amilyeneket Hollings vet fel a fejtegetéseiben. *(Hillary Rosen)*

A tartalmi ipar képviseletében a Disney és a Fox felkeresi a kongresszust, és azt mondja: „döntsetek ti… nektek kell megtalálni a megoldást". Azt hiszik, ez a „nyílt forrás". *(Maria Cantwell, az Egyesült Államok szenátora – D-Wash)*

Ezen kívül szükség van a régi autókat védő törvényre, amely kimondja, hogy az Egyesült Államokban eladott mozgó járművek mindegyikében régi autóalkatrésznek kell lennie. És vasútvédelmi törvény is kell, amely szerint nem árusítható tömegközlekedési jegy olyan szolgáltatásra, amely gyorsabb vagy kevésbé költséges a vonatozásnál. *(A. K. M. Adam tiszteletes)*

Képzeld el, mi lenne, ha a rádiózást előbb találták volna fel, mint a telefont. Az emberek azt mondanák: „Ugyan, kinek kell a személyes beszélgetés? A műsorszórásnál nincs jobb." *(Bob Frankston)*

Az engedély nélkül használható színkép (spektrum) a legjobb dolog, amit valaha tettünk. *(Michael Powell, az FCC elnöke)*

Senki nem jogosult a szabadság áldásaira, aki nem ügyel éberen annak megőrzésére. *(Douglas MacArthur)*

Azok, akik szabadalmi rendszerről beszélnek, többnyire érdekeltek is abban, és ezért kedvező színben tűntetik fel. Csakhogy a szabadalmak a sorsjátékokhoz hasonlatosak: az embernek csak ritkán származik belőlük előnye. A sorsjáték azt sugallja, hogy csak a nyerésre gondolj, a vesztésre soha, és a szabadalmaztatási rendszer is éppen ilyen. *(Richard M. Stallman)*

A megsebzett bika öklelésre kész bika. *(Jim Barksdale)*

A programozás olyan, mint a szex. Természetesen fizethetsz és nézheted a színészek játékát is, de bizonyos dolgok sokkal élvezetesebbek, ha magad csinálod. *(Miguel de Icaza)*

A Világháló egy felhízlalt program (bloatware). Annyi minden van rajta, hogy lehetetlen bármit is megtalálni. Gondolj bele, mekkora merevlemez-kapacitást foglal feleslegesen a sok különféle weboldal, amelyet mindössze a világ 0,00000000001 százaléka olvas el valaha. Mivel az emberek túlnyomó többsége csak a Yahoót, az Ebay-t vagy az MSN-t látogatja, nem lenne jobb a Világháló, ha csak Yahoo, Ebay és MSN lenne rajta? Mondhatni, „optimalizálttá" válna. *(Joel Spolsky)*

Linux Journal 2002. május, 98. szám

Linuxos színpadtechnika

A nyílt forráskód előnyei közé tartozik, hogy Linuxszal az általánosan megszokott felhasználási területektől merőben eltérő helyeken is találkozhatunk.

1. kép

2. kép

3. kép

4. kép

Hazánk fővárosa joggal lehet büszke színházaira, színművészeire, de természetesen hiba lenne megfeledkezni azokról a szorgos szakemberekről, akik munkája nélkül nem jöhetne létre az előadás – gondolok itt a díszletezőkre, az öltöztetőkre, a hang- és fénytechnikusokra. Abban a szerencsében részesültem, hogy hazánk egyik legismertebb befogadószínházának a kulisszái mögé tekinthettem be, amely akár hetente képes a nagyérdeműt új előadással meglepni. Gondolom, ebből a kis felvezetőből már sokan tudják, hogy a Thália Színházat próbáltam jellemezni. A művészbejárótól öltözőkkel szegélyezett folyosó visz a színpadra, ahol az ügyelőpult mellett méreteit tekintve egy laptopra emlékeztető grafikus terminál vonta magára a figyelmemet. Mint *Major Attila* színpadmestertől megtudtam, ez a szerény szerkezet a zsinórpadlás legtetején lévő kiszolgálógép egyik terminálja, és feladatát tekintve a díszletek mozgatását végzi – milliméter pontosan. Képes arra, hogy az előre megtervezett folyamatokat egy gombnyomásra véghezvigye. Tehát csak annyit tesz, hogy az adott pillanatban az egyik díszletet felemeli, például 4 méterre, miközben a másikat 6 méterre leengedi és így tovább. Viszont ezt bármikor hajlandó megtenni, óramű pontossággal. Talán nem is kell megemlítenem, hogy a feladat ellátására Linuxot használnak, ami egy kellemesen szerény, kimondottan a feladatra összpontosító grafikus, azaz X-felületen tart kapcsolatot a mesterrel. Természetesen igyekeztem kifaggatni Attilát, hogy miért pont linuxos gépet használnak. Nem volt más választási lehetőség, esetleg a vételár mértéke döntött a beszerzésnél? „Végtelenül egyszerű erre a magyarázat. Természetesen volt és van is más lehetőség, azonban nekünk az volt a legfontosabb, hogy olyan technikát vegyünk, ami hosszútávon is megbízható. Nem az ár döntött, hiszen egy ilyen technikában a program értéke a körülötte található berendezések árához viszonyítva elenyésző. Ő a mi kis cicánk, mert csak így becézzük, hiszen luxemburgi atyjától a CAT (Computer által támogatott színház) nevet kapta (➲ http://www.goodland.lu)."

Megsimogattam a cicát, majd új kísérőmmel, *Bodon Károly* úrral átsétáltunk a színpaddal szembeni legmagasabb páholyba, ahol régi színházi szokás szerint sötétített üvegtáblák mögé rejtve a világosítás vezérlőközpontja található.

A helyiségbe lépve egy hagyományos fénytechnikai vezérlőpulttal és négy karakteres színes monitorral találtam szembe magam. Az ADB vision 10/t rendszer Belgiumból, pontosabban Brüsszelből származik. Megalkotója, *Andrian Dee Bocker* (➲ http://www.adb.de) maga is gyakorló fővilágosító, és mivel többféle fénytechnikai pulttal is volt szerencséje dolgozni, úgy döntött, hogy olyan technikát készít, amely tökéletesen megfelel a kívánalmaknak, és megbízhatóság területén magasan túlszárnyalja elődeit.

A pult érdekessége, hogy ránézésre jól kezelhető, gondosan átgondolt kezelőfelületet látunk, ami a megszokott gombokat, potenciométereket tartalmazza, belül azonban egy teljesen hagyományos Pentium II-es alaplap végzi a háttérmunkát, illetve az asztal alatt a mentőegység (backup) található, mely első ránézésre egy szerény kis kiszolgálóra emlékeztet.

Bodon Károlyt arról faggattam, hogy mit is tud az ADB, mire használják, és miért éppen ezt alkalmazzák.

„Fontos megemlítenem, hogy az ADB-n Linux fut. Miért? Mert ennél megbízhatóbbat még nem sikerült találnunk. DMX 512 jellel kommunikál, ami 1990 óta elfogadott UISTT-szabvány. Tudom, ez így nem sokat mond, de a lényeg az, hogy 2×512, azaz 1024 csatornát tudunk használni, amin belül 60 jel áll rendelkezésünkre. Egy jel egy világosítási kép. Például egy pörgő-forgó lámpa átlagosan 8–40 jelet kap, így a nálunk található 22 awab motorikus kengyel és az 50 rolnis színváltót, meg az egyéb egységeket tökéletesen tudjuk működtetni (➲ http://www.adb.be)."

Szerettem volna a pultot a gyakorlatban is kipróbálni, de erre sajnos nem volt lehetőség, mivel éppen egy francia produkció színpadra állítása zajlott.

A háttérbe húzódva szemléltem, milyen hihetetlenül gyorsan programozzák az előadást. Már nem is csodálkoztam különösebben azon, hogy egy előadáshoz akár egy nap alatt képesek mindent beállítani, bár bonyolultabb produkciók esetében ez akár egy hetet is igénybe vehet. Melegséggel telt szívvel távoztam a színházból. A Linux ismét bebizonyította, hogy a nyílt, szabadon felhasználható forráskód előnyeit az élet szinte minden területén kamatoztathatjuk.

1. kép Központi kiszolgáló és a háttértároló
2. kép ADB vision 10/t világosító pult
3. kép Az ADB vision 10/t kezelőfelülete, mely a kézi vezérlést is lehetővé teszi
4. kép Beállító terminál, lényegében itt történik a csatornák kiválasztása

Gibizer Tibor
(gibzo@linuxmania.hu)
Újságíró, immár hét éve a Linux elkötelezett híve. Imádja a kutyákat, a kerékpározást és az autós csavargást.

Az LME tervei

A Linux-felhasználók Magyarországi Egyesületének elnöksége és az egyesület tevékeny tagjai töretlen lendülettel dolgoznak azon, hogy kedvenc egyesületünk lehetőségeihez mérten a legjobban feleljen meg azon követelményeknek, amellyek a hosszú távú eredményes munkához elengedhetetlenek.

Alig telt el pár izgalmakkal teli hónap, máris rendeződni látszik az egyesület működéséhez szükséges anyagi háttér. Ez elsősorban a 2000-ben indított, majd 2001-ben teljesített MeH (Miniszter Elnöki Hivatal) által életre hívott projekt végleges lezárásának köszönhető.

A MeH projekt jelen történetünket tekintve az első olyan vállalt kötelezettségünk volt, amely soha nem látott viharokat kavart a tagság körében. Talán kicsit felkészületlenek, gyakorlatlanok voltunk a feladat nagyságához mérten, azonban most, ennyi idő távlatából talán már kár újrarágni a csontot.

Egy egyesület működtetése néha komoly ügyviteli felkészültséget igényel. A folyamatosan változó állami szabályozásokkal lépést tartani még akkor sem könnyű, amikor tagjaink között olyan kiváló szakembereket tudhatunk, akik naponta szembesülnek a cégvezetés, a könyvelés stb. kínjaival. Elsősorban nekik köszönhető, hogy egyesületünkben rendeződnek a papírmunka azon területei is, amelyek eddig talán egy kicsit háttérbe szorultak. Ennek következménye, hogy már készülőfélben van a tartománykezelési szabályzat, illetve mivel az egyesületnek törvényes kötelessége, hogy számviteli politikája legyen, amelynek része a pénzkezelési szabályzat, elnökünk elvállalta, hogy elkészíti a tervezetet.

A Linux népszerűsítése egyesületünk tagjai számára mindig is rendkívüli fontossággal bírt. Ennek köszönhetően egymás után születnek a különböző linuxos honlapok, cikkeket írunk, linuxos tanácskozásokat szervezünk. Az elmúlt pár hónapban sikerült meghódítanunk egy eddigi fehér foltnak bizonyuló médiumot, a televíziót. Hosszas szervezés után végre megszületett hazánk első televíziós műsorsorozata a fix.tv-ben, amely minden szerdán délután 4 és 5 között Linuxszal foglalkozik .

Talán túl tág a „Linux népszerűsítése” meghatározás, véleményünk szerint ide azonban kívánkozik egy új próbálkozás, amely a „Gépfelszabadítás” nevet kapta.

A projekt célja, hogy olyan számítógépeket gyűjtsön össze, amelyek még használhatók, de mások, főként cégek leselejtezték. Ezeket a gépeket szeretnénk oly módon felszabadítani, hogy szabad, nyílt forrású operációs rendszerrel, azaz GNU/Linuxszal vértezzük fel őket. Az így újjáéledt gépeket iskolákkal, illetve olyan intézményekkel szeretnénk megpályáztatni, akik más módon csak komoly elkötelezettségek árán juthatnának olyan technikai berendezésekhez, amelyek például az oktatásban még kiválóan használhatók. Ezzel és a fentiekkel kapcsolatban örömmel lát minden hozzászólást, tanácsot, segítő kezet az LME.

Gibizer Tibor
(gibzo@linuxmania.hu)
Újságíró, mintegy hét éve a Linux elkötelezett híve. Imádja a kutyákat, a kerékpározást, az autós csavargást.

Digitális videokamera használata Linux alatt

Hogyan menthetjük le digitálisan tárolt filmjeinket a videokameráról? Ez bizony elég bonyolult folyamat, mégis mindenki egyszerűen megbirkózhat a feladattal, ha rendelkezik egy linuxos géppel, firewire kapuval és egy dv-csatlakozós videokamerával, no meg természetesen egy programmal, ami kezeli a két készülék közötti a kapcsolatot. A program neve `dvgrab`, ezt Debian/GNU alatt a `apt-get install dvgrab` paranccsal telepíthetjük.

A képlet egyszerű: dugjuk össze a két gépet egy erre megfelelő kábellel, állítsuk be a kamerát a megfelelő filmrészlethez. A következő paranccsal vehetjük rá a gépet, hogy a firewire kapuról érkező adatokat merevlemezünkre mentse:

```
dvgrab -format dv2 filcime
```

Ha a parancs kiadása után a kamerán elindítjuk a lejátszást, akkor az adatfolyam dv2-es formátumú AVI-ba kerül mentésre, a neve *filmcimexxx.avi* lesz (ahol a fájl sorszáma az xxx lesz). Ha másként nem rendelkezünk, az adatfolyam mindaddig rögzítésre kerül, amíg tart, megközelítőleg 1 GB-os darabokra szabdalva. Tapasztalatom szerint egy órányi anyag 13–16 GB közti helyet foglal a merevlemezen. Készüljünk fel, hogy ezt a rengeteg adatot csak gyors merevlemezek képesek hiba nélkül folyamatosan menteni. Ha merevlemezünk erre nem képes, a program önműködően képkockákat hagy ki, ami a filmben természetesen ugrásnak látszik. Nagyon vigyázzunk, hogy amikor a rendszermagot készítjük a feladathoz, semmiképpen ne kapcsoljuk be a FireWire menüpont alatt a *Többszörös ellenőrzés* menüpontját, ilyenkor ugyanis nagyon sok adatot ír pluszba a `syslog` a fájljaiba, ami szintén sebességcsökkenéshez és hibákhoz vezet.

Azért a dv2-es formátumot ajánlom, mivel elég nagy fájlok jönnek létre, és ezt a formátumot a `mencoder` kiválóan kezeli, így az 1 GB-os fájlokat kedvünk szerint tömöríthetjük. Egyórányi anyag általában kényelmesen és nagyon jó minőségben elfér egy 650–700 MB-os CD-n.

Csontos Gyula

Beágyazott Rendszerek Konferencia (ESC) 2002

ESC, San Francisco – Rick szerint a beágyazott Linux többé már nem csak a fura újoncok egyike.

A március elején San Franciscóban megrendezett Embedded Systems Conference (ESC – Beágyazott Rendszerek Konferencia) rendezői a beágyazott rendszerek piacának különféle szereplői közül mintegy 15 ezer látogatóra számítottak. Ugyan a gazdasági körülmények a vártnál rosszabbul alakulnak, mégis körülbelül 12 ezer érdeklődő jelent meg. A mérsékelt figyelem ellenére az idei, a Moscone Convention Center két hatalmas termét elfoglaló ESC az eddigi legnagyobb volt.
A beágyazott Linux oldaláról nézve a legfontosabb hír az, hogy a beágyazott Linux többé nem számít újdonságnak! Immár teljes joggal foglal helyet a három legfontosabb és nélkülözhetetlen beágyazott operációs rendszer között, amelyet gyakorlatilag minden beágyazott vasnak (lapkáknak, áramköröknek, teljes rendszereknek), középrétegnek (middleware), alkalmazásnak és eszköznek támogatnia kell. A másik kettő általában a VxWorks és valamelyik beágyazott Windows-változat. A beágyazott Linux, a VxWorks és a beágyazott Windows mellett az egyéb beágyazott operációs rendszerek a jelek szerint a „futottak még" kategóriába kerültek, és csak akkor rúghatnak labdába, ha egy nagyobb vásárló hajlandó a fejlesztést megfizetni, vagy nagy mennyiségű terméket vásárolni. Az alábbiakban néhány érdekességet szeretnék kiemelni a 2002-es, San Franciscóban megrendezett ESC kiállításról – a beágyazott Linux szemszögéből.

Az Embedded Linux Consortium kiadja az Embedded Linux Platform szabványt

Az Embedded Linux Consortium műszaki nyílt összejövetelt tartott március 12-én, amelyen a kétéves csoport kilépett eredeti, általános célú, ugyanakkor szabványos beágyazott Linux létrehozásának szorgalmazására (ELC Platform Specification) kiterjedő feladatköréből. Több mint 125 érdeklődő tűnt fel, és bár nem volt névsorolvasás, a kérdőíveken a világ legnagyobb és legbefolyásosabb program-, félvezető és elektronikai fejlesztő cégeitől – HP, Hitachi, IBM, Intel, Microsoft, Motorola, Panasonic, Samsung, Sharp, Sony, Texas Instruments, Toshiba és Wind River – érkezett képviselők nevei tűntek fel. Egyes megfigyelők panaszai szerint az ELC Platform Spec. fejlesztése túl lassan halad ahhoz, hogy a piacon is hasznosítani lehessen, mások szerint viszont örömteli, hogy végre létrejött egy alapítószervezet (az Intellectual Property Agreement avagy IPA), amelynek révén a nagy cégek is részt vehetnek az előírások fejlesztésében. Az ELC eredeti szabályai akadályozták a szabványfejlesztési tevékenységet, így nélkülözhetetlen volt valamilyen formába önteni őket, hogy megváltozhasson a csoport alapszabálya.
Az ELC találkozó két vezető felszólalása a hálózaton is elérhető.
Az ELC létrehozta első, az ELC Platform Spec. kidolgozásával megbízott munkacsoportját, és várja azokat az önkénteseket, akik részt szeretnének venni a 2002 végére kiadandó előírás-gyűjtemény kidolgozásában. Az ELC (➲ http://www.embedded-linux.org) további munkacsoportok létrehozását is indítványozta, amelyek érdeklődési köre a valós idejű Linux, a biztonság, a vezeték nélküli API-k, a magas rendelkezésre állás, az eszközillesztők és számos egyéb témakör lenne.

Rövid séta az ESC Vendor Expón

Az ESC útmutatója szerint a beágyazott Linux kategóriába összesen 39 vállalkozás sorolta magát. Az alábbiakban a kiállításon szereplő számos beágyazott linuxos témájú bemutató, műszaki megoldás és termék közül szeretnék megemlíteni néhányat.

- ***Lineo***
  Sajtókonferenciát tartottak, amelyen számos új set-top berendezést, otthoni átjárók (residental gateway) és zsebtitkárok fejlesztésével kapcsolatos együttműködési megállapodást jelentettek be. Érdekes bejelentés volt az új Linux-alapú zsebtitkár (a Kaii, nemrég mi is írtunk róla), amelyet Indiában fejlesztenek, és amelynek programkészlete nagymértékben hasonlít a Sharp Zauruséra. A Lineo épp egy fontos, termékeinek új elhelyezését eredményező, elsősorban három kiemelt piaci részt – kézikészülékek, digitális média és otthoni átjárók – megcélzó átállást hajt végre. Ez az irányzat tükröződött a cég standján tartott bemutatókon is, amelyek inkább egy-egy szűkebb feladatkörre, mint általános célokra alkalmas eszközökről szóltak. ➲ http://www.lineo.com
- ***LynuxWorks***
  Bejelentették a LynxOS 4.0-s változatát, amely állítólag megfelel a Linux ABI-követelményeknek. Ez azt jelenti, hogy a LynxOS, amely egy a Linuxszal API-szinten eddig is sok tekintetben együttműködni képes (tehát egyes programokat újrafordítás után futtatni lehetett rajta), mégis zárt valós idejű operációs rendszer (RTOS) volt, most már módosítás nélkül is képes bizonyos Linux bináris állományokat futtatni. A bemutatón láthattuk, hogy LynxOS rendszeren módosítás nélkül fut az Opera böngésző és a Quake. Megjegyzendő, hogy ehhez az szükséges, hogy a Linux-alkalmazások dinamikusan csatolt könyvtárakat hívjanak meg, ugyanis a `glibc` könyvtárak különleges változatait kell használni. A LynuxWorksnél az ilyen programokat „jó magaviseletű" programoknak nevezik, és emlékeztetnek rá, hogy az LSB (Linux Standard Base) is a könyvtárak dinamikus hívását írja elő.
  ➲ http://www.lynuxworks.com
- ***MontaVista Software***
  Szokás szerint méretes standot állítottak fel, amelyen számos eszközt és programot ismerhettünk meg, köztük a MontaVista High Availability Frameworköt; a Visual Age Micro Edition Java VM-et, amely egy Qt/Embedded alapú Java AWT-vel egészült ki; egy az IBM PowerPC 405GP processzorára épülő set-top-box hivatkozási készüléket; láthattunk Sharp Zaurus zsebtitkáron futó

MontaVista Linuxot, valamint szemügyre vehettük a MontaVista Linux 2.1-es változatának valós idejű szolgáltatásait és a támogatott processzorokat. Igazán érdekes bemutató volt a valódi, Linuxot futtató üdítőautomata (USA Technologies ePort), amelynek egy RadiSys StrongARM-alapú, beágyazott, egylapkás, MontaVista Linuxot futtató gép képezte a lelkét. A MontaVista bejelentette, hogy a Panasonic Digital Concepts Center (PDCC) – a Matsushita Electric Industrial Co., Ltd tulajdona – jelentős beruházással támogatja a céget.
➲ http://www.mvista.com

- ***Red Hat***
  Bemutatták beágyazott programcsaládjukat (Embedded Linux, eCos, RedBoot, GNUPro eszközök), valamint néhány érdekes termékkel ismerkedhettünk meg, köztük egy eCos-alapú Brother nyomtatóval, a Symbol Wireless vonalkódolvasóval (Linux-alapú), a Rymic mostoha körülményekhez tervezett autós számítógépével (linuxos), az Ericsson Screenphone (linuxos) és a Sony PS2 (GNUPro) készülékével, az Intel otthoni útválasztójával (Linux), az Iomega Hip Zip (eCos) nevű újdonságával és a Delphi autós telematikai rendszerével (eCos). Figyelmemet különösen az a bemutató ragadta meg, amely a Red Hat egyik emberének, *Clark Williamsnek* nemrég elvégzett vizsgálati eredményét szemléltette. Williams két elterjedt, a Linux-rendszermag késleltetéseit csökkentő módszert hasonlított össze: a MontaVista által bevezetett időosztásos foltot és az alacsony késleltetésű foltot, amely *Ingo Molnar* munkáját dicséri. Hogy melyik a jobb? Mindkettő jó, a részletes leírás elérhető az Interneten, a ➲ http://www.linuxdevices.com/articles/AT8906594941 és a ➲ http://www.redhat.com/embedded címen.
- ***REDSonic***
  Bemutatták RED-Builder nevű, jó néhány eszközt támogató, grafikus, beágyazott Linux képfájlkészítő és összevont fejlesztői környezetüket (IDE), amellyel egy kisebb Linux-rendszert egyetlen kattintással létre lehet hozni és a célkészülékre lehet tölteni, még hibakeresésre is képes. A célkészülék Ampro Encore 500 (x86), ITE 8152EVB (StrongARM) vagy MIPS 32-bites Malta Board-alapú lehet. A REDSonic elhozta SecureSOHO nevű programkészletét is, amellyel még csekély számítási képességű Linux-alapú átjáró- és tűzfalkészülékek is kiterjedt hálózati és biztonsági szolgáltatásokat nyújthatnak, miközben könnyen kezelhető felhasználói felületen keresztül felügyelhetők és állíthatók be.
  ➲ http://www.redsonic.com
- ***TimeSys***
  A kedvezőtlen gazdasági körülmények ellenére több jó hírrel érkeztek a kiállításra. Bejelentették, hogy 15,5 millió dollár támogatáshoz jutottak, és sajtóközleményük is büszkén hirdeti biztos helyzetüket: „A beágyazott rendszerprogramok úttörője megerősítette vezető helyét az iparágban". Standjukon négy, fejlesztői csomagokkal kapcsolatos bemutatót láthattunk, amelyek a TimeSys Linux/GPL Embedded Linux-terjesztésével és a hozzá fűződő eszközökkel, Windows-rendszerekkel, valós idejű kiterjesztésekkel, szolgáltatásminőséget garantáló CPU-foglalásokkal és valós idejű hálózatkeze0léssel kapcsolatosak, természetesen eltérő felépítésű processzorokon (x86, ARM, MIPS, PowerPC, UltraSPARC, XScale, SuperH) és gépeken. A cég számos új fejlesztői csomagot is bejelentett, és a jövőre nézve is a megjelenések lendületes folytatását ígéri.
  ➲ http://www.timesys.com

A beágyazott Linux mindenütt jelen van

Az Addison-Wesley új beágyazott Linuxszal kapcsolatos könyvet adott ki. *Dr. Craig Hollabaugh* „Embedded Linux: Hardware, Software and Interfacing" című 432 oldalas műve a különféle kapcsolattartásra használható alkalmazások fejlesztését és megvalósítását taglalja beágyazott Linux-alapú készülékeken. Hollabaugh szerint a könyv a „Trailblazer" tervezetet is ismerteti, amely egy képzeletbeli téli menedék önműködővé alakítását célozza – a leírás a kezdeti alapozástól és tervezéstől kezdve a megvalósításon keresztül egészen a rendszer egybeépítéséig terjed. Végigkövethetjük, hogy a Trailblazer-mérnökök hogyan választják ki a céleszközöket, hogyan hozzák létre a fejlesztői környezetet, végzik el a szükséges adatok begyűjtését, építik fel a vezérlést és a multimédiás eszközöket, majd hogyan készítik el az eszközillesztőket és az egységes programkódot.
A Lineo ismét szorosabbra húzza a nadrágszíjat. A gazdasági folyamatok kedvezőtlenre fordulására hivatkozva a Lineo márciusban 138-ról 75–80-ra csökkentette alkalmazottainak létszámát, mondta el *Matt Harris* elnök. Harris szerint a Lineo folytatja a múlt ősszel elkezdett folyamatot, amelynek során három kiemelt piaci részre összpontosítja erőit: kézi számítógépek (zsebtitkárok és intelligens telefonok), peremkészülékek (otthoni átjárók, tűzfalak, útválasztók) és digitális tévék (set-top boxok és szórakoztató központok). Múlt év szeptemberében a Lineo bejelentette, hogy 60 alkalmazottat bocsát el, és további 100 embertől vált meg, így mindössze 110 foglalkoztatottja maradt. Korábban, 2001 júniusában is volt elbocsátás, amikor a létszám 322-ről 280-ra csökkent. Harris reményei szerint a legutóbbi lépések a céget 2002 közepére nyereségessé teszik.

Linux Journal 2002. június, 98. szám

Rick Lehrbaum
(rick@linuxdevices.com)
hozta létre a LinuxDevices.com „beágyazott Linuxok portálját", amely nemrég tagja lett a ZDNet Linux Resource Centernek.

Jogos vagy jogtalan?

Idén április 1-jétől hatályos a büntető törvénykönyv 1978. IV. cikkelyének a Számítástechnikai rendszer és adatok elleni bűncselekményről szóló passzusa. Elsőként idézzük fel magát a törvényt:

Számítástechnikai rendszer és adatok elleni bűncselekmény

300/C. §

(1) Aki számítástechnikai rendszerbe a számítástechnikai rendszer védelmét szolgáló intézkedés megsértésével vagy kijátszásával jogosulatlanul belép, vagy belépési jogosultsága kereteit túllépve, illetőleg azt megsértve bent marad, vétséget követ el, és egy évig terjedő szabadságvesztéssel, közérdekű munkával vagy pénzbüntetéssel büntetendő.

(2.) Aki

a) számítástechnikai rendszerben tárolt, feldolgozott, kezelt vagy továbbított adatot jogosulatlanul megváltoztat, töröl vagy hozzáférhetetlenné tesz,

b) adat bevitelével, továbbításával, megváltoztatásával, törlésével, illetőleg egyéb művelet végzésével a számítástechnikai rendszer működését jogosulatlanul akadályozza, vétséget követ el, és két évig terjedő szabadságvesztéssel, közérdekű munkával vagy pénzbüntetéssel büntetendő.

(3.) Aki jogtalan haszonszerzés végett

a) a számítástechnikai rendszerbe adatot bevisz, az abban tárolt, feldolgozott, kezelt vagy továbbított adatot megváltoztatja, töröl vagy hozzáférhetetlenné tesz, vagy

b) adat bevitelével, továbbításával, megváltoztatásával, törlésével, illetőleg egyéb művelet végzésével a számítástechnikai rendszer működését akadályozza és ezzel kárt okoz, bűntettet követ el, és három évig terjedő szabadságvesztéssel büntetendő.

(4.) A (3.) bekezdésben meghatározott bűncselekmény büntetése

a) egy évtől öt évig terjedő szabadságvesztés, ha a bűncselekmény jelentős kárt okoz,

b) két évtől nyolc évig terjedő szabadságvesztés, ha a bűncselekmény különösen nagy kárt okoz,

c) öt évtől tíz évig terjedő szabadságvesztés, ha a bűncselekmény különösen jelentős kárt okoz.

Ennyi a törvényszöveg, amely informatikus körökben oly nagy vihart kavart, és rácsodálkoztunk: hogy is van ez? Létrejött egy büntetőtörvényi alakzat, mely korábban nem ismert, vagy ismert, de a büntető törvénykönyvben korábban nem szerepelt cselekményeket a büntetőjog hatálya alá vont, és azokat bűncselekményeknek, enyhébb esetben vétségnek, súlyosabb esetben bűntettnek minősít és hozzá büntetési tételeket rendel.

A kérdés az, kinek milyen cselekménye milyen tényállást valósít meg, és kit miért ítélhet el a független bíróság? Hogyan kerülhető el, hogy szakmai döntéseinket mások bűncselekményként értékeljék és akár alaptalanul is, de büntetőeljárást kezdeményezzenek ellenünk?

Írássorozatomban ezekre a kérdésekre keresek részben gyakorló informatikusként, részben joghallgatóként – jogi tanulmányaim alapján – választ.

Előre kell bocsátanom, írásom inkább elgondolkodásra serkentő céllal íródott. Kapcsolódó jogi fejtegetéseim tartalmát eddig végzett tanulmányaim keretei adják, így azok felhasználhatósága gondolatébresztésen kívül másra nem terjedhet ki, és egy új, eddig ismeretlen határterületen bolyongunk majd: a jog és az informatika határán.

A törvényszöveget elsőre elolvasva – valljuk be – rémület lesz úrrá rajtunk. Ha csak a jelzett jogszabályhelyet nézzük, látszólag szándék kérdésévé válik, milyen szakmai eljárás minősíthető jogellenesnek.

Nézzük meg először; mivel vagyunk fenyegetve a büntetőjog oldaláról! Amit e törvényhely szerint elkövethetünk, az bűncselekmény – tehát az a cselekmény, amely szándékosan, vagy ha a törvény a gondatlan elkövetést is bünteti, gondatlanságból kerül elkövetésre, és amely veszélyes a társadalomra, és amelyre a törvény büntetés kiszabását rendeli.

Egyidejűleg kell fennállnia

- társadalomra való veszélyességének,
- az elkövetői szándékosságnak vagy a büntetni rendelt gondatlanságnak,
- a cselekmény büntetni rendeltségének – tényállás megvalósulásának
- és a bűnösségnek.

A cselekmény a törvény értelmezésében megvalósulhat tevéssel, tehát valamilyen aktív cselekedettel, de valamitől való tartózkodással, azaz nem tevéssel is.

Ezek az alapfogalmak jelen vizsgálódásunkban.

Az informatikai munka során cselekményeink (feladataink) vannak: az adatbiztonság megteremtése a működés fenntartása érdekében, de cselekmény egy folyamat figyelemmel kísérése is, jóllehet mi magunk tevőlegesen nem avatkozunk be. Ezen cselekmények a büntetőjogban akként kerülnek megvizsgálásra, hogy cselekményünk megfeleltethető-e valamely törvényi tényállásnak, eleget teszünk-e a vonatkozó szakmai előírásoknak, illetve mi személy szerint megtettük-e mindazt, ami az adott helyzetben elvárható.

Ezen cselekmények megítélése a jelen törvénymódosítás hatályba lépésétől többé már nem szakmai, hanem adott esetben büntetőjogi kérdés lett.

Az adott helyzetben elvárható és a szakmai szabályoknak megfelelő cselekmények adják az informatikai szakma felelősségének alapjait.

E felelősség a jog több oldaláról is behatárolt:

- büntetőjogi – e törvény cikkelyein alapulva;
- polgári jogi;
- munkajogi;
- szabálysértési, amennyiben ezt rendelet vagy önkormányzati rendelet szabálysértéssé nyilvánítja.

A büntető törvénykönyv tárgyalt értelmezési és alkalmazási gondjai szerintem innen (is) kezdődnek.

Büntetőjogi oldalról szemlélve informatikai munkánk

során a számítástechnikai rendszer és adatok elleni bűncselekményt – jelen törvényi szabályozás szerint – szándékosan lehet elkövetni. Szándékosan követi el a bűncselekményt, aki magatartásának következményeit kívánja vagy e következményekbe belenyugszik.
Ez viszonylagosan jól érthető és alkalmazható mindennapi élethelyzeteinkre: ha valakik szándékosan tönkretesznek például egy adatállományt, amellyel a munkáltatónak, megbízónak kárt okoznak, azért e törvényhely szerint felelniük kell. A lényeges részlet tehát a célirányos szándékos károkozás.
A szándékosság megvalósulhat például egy merevlemez szándékos megrongálásával, törlésével, de megvalósulhat a beavatkozástól való tartózkodással, ha az károkozás előidézésére irányul. Így van, ha például egy egyre többször eszközhibát jelző meghajtót azért nem mentenek (cserélnek) le, mert tudják, hogy e folyamat végén tönkremegy, és ezzel akarnak kárt okozni; tehát a hiba keletkezése és a kár bekövetkezte között a „nem tevő" magatartás oksági kapcsolatot teremt – bizony, ez is szándékos magatartásnak minősül. A kárt a szándékos célirányos magatartás okozta, még ha ez tartózkodásban valósult is meg.
E törvény 300/C § (4) alkalmazásában jogtalan haszonszerzés végett meghatározott esetekben az érték, a kár, a vagyoni hátrány, a mérték összege, illetőleg az adó-, a járulék, a magánnyugdíj-pénztári tagdíj bevételcsökkenésének összege

- *kisebb* a kár, ha tízezer forintot meghalad, de kétszázezer forintot nem halad meg,
- *nagyobb*, ha kétszázezer forintot meghalad, de kétmillió forintot nem halad meg;
- *jelentős*, ha kétmillió forintot meghalad, de ötvenmillió forintot nem halad meg;
- *különösen nagy*, ha ötvenmillió forintot meghalad, de ötszázmillió forintot nem halad meg;
- *különösen jelentős*, ha ötszázmillió forintot meghalad.

A kármérték összege az informatikában összetett fogalom: egyrészt tartalmaznia kell az eredeti állapothoz képest bekövetkező értékcsökkenést (mely sokszor az eredeti állapot helyreállítási költségeiből állapítható meg), az elmaradt hasznot, és a felmerült vagyoni és nem vagyoni értékcsökkenést, hátrányt.
Gondoljunk bele, egy üzemelő gyár számítógépes gyártásirányítási rendszerének vagy egy bank központi számítógépének tudatos tönkretételével milliárdos károkat lehet okozni! Elég, ha a kiesett forgalomra, az adatállományok helyreállításának költségére (ha egyáltalán lehetséges), vagy egy bank esetében az ügyfelek bizalomvesztésére gondolunk. Hatalmas az informatikai szakma felelőssége. Ezek szerint bűnöző lenne az a rendszergazda, akinek rendszere tönkremenetele során kára keletkezik?
Ez így nem lenne igazságos. A törvényalkotó figyelmes volt a jogszabály megalkotásánál. A szándékosan elkövetett cselekményeket emelte be a büntetőjogba. Jelen szabályozás alapján a gondatlanságból (tudatos gondatlanságból és hanyagságból) elkövetett cselekmény nem minősül bűncselekménynek, legyen az eredményében akár a szándékos bűncselekménnyel azonos eredményű. Ennek megértéséhez ki kell fejteni a *tudatosság* és a *szándékosság* közti különbséget. Tudatos minden olyan körülmény, amelyről tudunk, ismeretünk van róla, értelmünk átfogja. A szándékosság ennél több, mivel egyrészt tudatos (mert az általunk ismert körülményeken nyugszik), másrészt ennél is több, mert szándékosság esetén az elkövető tudata átfogja a cselekmény által elérni kívánt célt-eredményt is.

Btk. 300/C:
(2.) Aki
a) számítástechnikai rendszerben tárolt, feldolgozott, kezelt vagy továbbított adatot jogosulatlanul megváltoztat, töröl vagy hozzáférhetetlenné tesz,
b) adat bevitelével, továbbításával, megváltoztatásával, törlésével, illetőleg egyéb művelet végzésével a számítástechnikai rendszer működését jogosulatlanul akadályozza, vétséget követ el, és két évig terjedő szabadságvesztéssel, közérdekű munkával vagy pénzbüntetéssel büntetendő.

Ez nem jelenti azt, hogy aki gondatlanságból követi el károkozó cselekményét, az mentesülne a felelősségre vonhatóság alól. Nem, csak rá a felelősség más, például polgári jogi, munkajogi eseteit kell alkalmazni – kártérítési felelőssége ugyanúgy fennáll.
Gondatlanságból követi el a (bűn)cselekményt az, aki előre látja magatartásának lehetséges következményeit, de könnyelműen bízik azok elmaradásában; úgyszintén az is, aki e következmények lehetőségét azért nem látja előre, mert a tőle elvárható figyelmet vagy körültekintést elmulasztja.
A gondatlan magatartás akkor válik bűncselekménnyé, ha a Btk. Különös Része az adott tényállás gondatlan alakzatát is büntetni rendeli. Jelen esetünkben az informatikai cikkelyek nem tartalmaznak gondatlan tényállást, a gondatlan elkövetés itt tehát nem bűncselekmény (azonban a kártérítési felelősség polgári jogi vagy munkajogi úton ekkor is megállapítható).
Következésképpen az informatikai károkozásnál a felelősség megállapításánál általánosságban és nem büntetőjogi értelemben kell objektíve (szakmai írott és íratlan szabályok) és szubjektíve az adott helyzetben elvárható magatartás tanúsításának kötelezettségére alapozott felelősségről beszélnünk.
A gondatlanságnak két alakzata van: az egyik a tudatos gondatlanság, a másik a hanyagság. Tudatos gondatlanság esetén az elkövető előre látja magatartásának következményeit, de könnyelműen bízik azok elmaradásában. Ez különösen akkor állapítható meg, ha e bizalmát valamire alapozni tudta, például bízik a saját ügyességében, vagy valamilyen külső okban, ami

megakadályozhatta volna a következményeket.
Hanyagság esetén az elkövetői oldalon a cselekmény előrelátása mint tudati elem és értelemszerűen a cselekményhez kapcsolódó érzelmi (lelki) elem is hiányzik. Nem törődik a várható következményekkel és nem is érdekli azok lehetséges kimenetele.
Nagyon fontos minőségi elhatárolást húzni a szándékos mulasztásban megnyilvánuló bűncselekmény (e törvény szerint büntetendő) és a jelen szempontból bűncselekménynek nem minősülő tudatos gondatlanság között. Tudatos gondatlanság esetén felmerül a várható káros kimenetel lehetősége, de bízom benne, hogy az vagy nem következik be, vagy bízom az ügyességemben, hogy képes leszek elhárítani.
Szándékos mulasztásban megnyilvánuló bűncselekmény esetén a várható káros kimenetel elérésére törekszem azáltal, hogy nem teszek semmit. Hagyom, hogy a folyamat úgy okozzon kárt, hogy nekem ne kelljen tevőlegesen közreműködnöm. A következő példával szeretném a különbséget megvilágítani.
A számítógépes rendszerben egy ismert számítógépes vírus jelenik meg, és nem irtják ki haladéktalanul, hanem megvizsgálják, milyen tüneteket okoz, hogyan deríthető fel, mit tapasztalható vele kapcsolatban, tehát abban bíznak, hogy a vírus későbbi kiirtása is lehetséges. Tudatos gondatlanság esetén az illető bízik benne, hogy a rendszer használhatatlanná válása előtt vírusirtóval majd megfékezi a vírus terjedését, ám ha ez valamilyen oknál fogva mégsem sikerül, és a rendszer tönkremegy vagy működése akadályozott lesz, máris kész a tudatos gondatlanság esete.
A szándékos mulasztásos bűncselekmény esetén az elkövető tud a vírus megjelenéséről – adott esetben akár maga telepítette –, csakhogy nem avatkozik be, mert azt akarja, hogy a vírus elterjedve kárt okozzon.
Egyes vírustípusoknál az illető egyéb műveletek (végtelen ciklusos lemez vagy memóriaművelet) végzésével a vírus fajtájától függően a számítástechnikai rendszerben tárolt, feldolgozott, kezelt vagy továbbított adatot jogosulatlanul megváltoztatja, törli vagy hozzáférhetetlenné teszi, a rendszer működését egyéb művelet végzésével jogosulatlanul akadályozza (a rendszert lelassítja, lassúságával üzemi használatra alkalmatlanná teszi).
A kérdés a szöveg olvastán magától adódik: lehet-e egy rendszer működését jogosultan akadályozni? A válasz kézenfekvő: természetesen igen, gondoljunk csak a háttérben futó mentésre egy kiszolgálón, amelyet a jogosult rendszergazda indított el, és amely adott esetben az ügyfelek hangos tiltakozását okozza a rendszer lassúsága miatt. Még jó, ha nem ragadtatják magukat súlyosabb cselekményre! Igaz, itt felvetődik a szakmai kérdés: mikor kell ezeket a kapacitásigényes eljárásokat végezni, azaz rendelkezik-e róla a belső szabályzat vagy szokás, illetve mit mond a szakmai etika: illik-e feltartanom a többieket? A szolgáltatásnak az elvárt szinten kell működnie – ez a szakmai etika alapkövetelménye.
A törvényszöveg kapcsán felvetődő másik kérdés, hogy ki és mire jogosult? Ennek megválaszolásával azonban majd legközelebb foglalkozunk.

Granek István
Elektronikai műszerész, kajak-kenuedző és immár Linux-rajongó is. 1998 óta helyi hálózatok tervezésével és kiszolgálóépítéssel foglalkozik, valamint a JPTE harmadéves joghallgatója. Szeret síelni, biciklizni, úszni – és még keresi az igazi nőt.

Netcraft

A Netcraft webkiszolgáló-felmérése rendszerint örömteli hírekkel szolgál a Linux hívei számára, ám a legutóbbi, 2002. márciusi jelentésről ez nem mondható el.
A ➲ http://www.netcraft.com/survey weboldalon található grafikonokon látható az a küzdelem, amely az Apache és a Microsoft IIS webkiszolgálói között folyik 1998 óta. 2001-ben az IIS jelentősen csökkentette lemaradását az Apache-hoz képest, amely a webkiszolgálók piacának több mint hatvan százalékát uralta, végül azonban úgy tűnt, hogy 2002 februárjában a folyamat a visszájára fordul. Ám márciusban az IIS részesedése 4,89 százalékkal nőtt, míg az Apache-é 4,67 százalékot esett, így a fejlesztők ranglistáján (Top Developers) 57,36 százalékon áll az Apache és 34,02 százalékon az IIS.
A Netcraft szerint ez azt jelenti, hogy közel kétmillió webhelyen történt váltás „elsősorban annak eredményeként, hogy a register.com és a Network Solutions tartományparkoló szolgáltatása windowsos kezelőfelületre költözött. A Register.com szolgáltatása eddig Apache kiszolgálón és Linux-rendszeren futott, jelenleg azonban a Windowsra való áttérés zajlik" (idézet a Netcraft jelentéséből).
A leparkolt tartományok szolgáltatásai természetesen sokban elmaradnak a tevékeny tartományokétól, így ez az adat a kiszolgálói oldalról nézve valószínűleg nem mond eleget a Világháló valós gyakorlati felhasználásáról. Mindazonáltal bosszantó, ha csak pár százalékponttal is kevesebbel dicsekedhetünk.
Talán ha befejeződik a nagy tartománynév-kibocsátók átállása a Sötét oldalra, az Apache részesedésének gyengülése is megszűnik, sőt, akár meg is fordul.

Doc Searls

Linux Jorunal, 2002. május, 97. szám

Honosítások

Beszélgetés Fejős Tamással, az LME újonnan megalakult Honosítás csoportjának vezetőjével.

Hosszú ideje vesszőparipám a honosítások témaköre. Egy évvel ezelőtt azt írtam, hogy a GNU/Linux hazai elterjedésének egyik legfontosabb akadálya, hogy még nem lehetséges összerakni olyan irodai rendszert, amelyet egy angolul nem tudó titkárnő is használni tudna. Bár lassan és amolyan magyar módra, de sokat fejlődtünk. Elmondhatjuk, hogy van magyar grafikus munkakörnyezet, szövegszerkesztő, böngésző, sőt, helyesírásellenőrző rendszer is. Most, hogy itt van az ajtóban az UHU Linux, ami (remélhetőleg) minden fontosabb területen magyarul szól a felhasználóhoz, felcsillant annak reménye, hogy a teljes rendszer a mi nyelvünket beszélje. Miért mondtam, hogy „magyar módra"? Mert hogyan képzeli el egy reménykedő ember, miként is zajlik e programok fordítása? Alakul egy központi csoport, amelyik összegyűjti a szakkifejezéseket, elérhetővé teszi őket, összefogja a fordítókat, közös felületet teremt, és nem utolsósorban szakembereket, nyelvészeket kér fel a munkában való részvételre. Ezután egyes kisebb csoportok nyugodtan elkezdhetnek fordítani egy-egy programot vagy alrendszert, azzal a biztos tudattal, hogy a kész fordítás illeszkedik az egész rendszerbe.

No, nálunk ez nem teljesen így történt. Mivel nem volt egy központi, mindenki számára elérhető és használható fordítástámogató központ, több termék magyarításához külön csoportok jöttek létre, sőt, a magyar nyelvű programleírásokat kinek-hogy-tetszik alapon kezdték fordítani. Az áldatlan állapot megszüntetésére több lista és fórum kelt életre. Az LME is létrehozta a *magyar* nevű levelezőlistát, ahol a fordítók szavakat, kifejezéseket vitathatnak meg. Idővel azonban bebizonyosodott, hogy nem elég egy ilyen listát üzemeltetni, működő és használható „szervezetre" van szükség.

Fejős Tamás még a tavasszal történt LME elnökségválasztáson elmondta, hogy milyen tervei vannak, és végül június 12-én hivatalosan is megalakult az *LME Honosítás csoport*. A csoport terveiről, rövid- és hosszú távú céljairól beszélgettem Tamással.

Szy György: *Sok csoport jött már létre fordítások támogatására, miben jelent újat a tiétek?*

Fejős Tamás: A jelenlegi állapot egyik legnagyobb rákfenéje, hogy a kisebb csoportok nem tartják a kapcsolatot, így egymástól függetlenül sokszor azonos gondokkal küszködnek. Az *LME Honosítás* nem maga szeretné a munkát irányítani, sokkal inkább olyan hátteret kíván megteremteni, amit az összes fordítással, honosítással foglalkozó csoport használni tud. Tudjuk, hogy ez nehéz feladat, hiszen az egyes csoportok az általuk kialakított környezetet használják, ezeket a környezeteket pedig igen nehéz összehangolni. Gondoljunk csak a szóhasználatra – igény van egy központi szó- és kifejezéstárra, amelyet mindegyik csoport elfogad. Bár még menthető a helyzet, egy ilyen központi adattár kialakítása (és főleg elfogadtatása) komoly kihívás.

Sz. Gy.: *A csoport tehát szervezőként, irányítóként kíván tevékenykedni?*

F. T.: Igen, de nem kizárólag. Elsősorban az Egyesület eszközeit, szolgáltatásait igyekszik elérhetővé tenni a fordítók számára. Gondolok itt olyan lehetőségekre, amelyeket egy Egyesület könnyebben megold; támogatások, hivatalos ügyek intézése, pénzügyi háttér, eszközök stb. Nem az a célunk, hogy egy rendszert „felülről" kényszerítsünk rá a fordítókra, hanem hogy a már tevékenykedő csoportok számára hozzuk létre a továbblépéshez szükséges hátteret.

Sz. Gy.: *Milyen rövidtávú céljaitok vannak?*

F. T.: A legégetőbb talán a szóhasználat egységesítése. Itt nemcsak szótár létrehozására (a meglévők összeolvasztására) gondolok, de a fordítók közötti párbeszéd elősegítésére is. Szerencsére több olyan terv is van, amelynek rendkívül örülünk. Ilyen például az FSR2 *Magosányi Árpád* irányítása alatt. Jelenleg a kapcsolatok felvétele folyik (például az fsf, a KDE vagy a Gnome magyarítással foglalkozó embereivel és a hasonló szervezetekkel).

Ahogy mondtam, elsősorban nem a fordítások elvégzése a célunk, hanem a támogatás, ezért is keressük a kapcsolatokat, de nyitottak vagyunk munkák, tervek közös szervezésére, lebonyolítására is. Ez azokra a csoportokra is vonatkozik, amelyek nincsenek kapcsolatban az LME-vel, hiszen elvünk, hogy nem az LME, hanem a GNU/Linux érdekeit nézzük. A fordítások közül jelenleg az OpenOffice.org csomaghoz és a Mozilla böngészőhöz tartozó súgó fordítását tartjuk a legfontosabbnak.

Sz. Gy.: *Mindig fontos kérdés a pénz. Ti hogyan álltok ezen a területen?*

F. T.: Ahogy mondtam, célunk a fordítókat ügyintézéssel és olyan tevékenységgel segíteni, amit az LME keretein belül könnyebben megtehetünk. Az Egyesület rendelkezik anyagi kerettel, bár a források természetesen korlátozottak. Reményeink szerint támogatások, pályázatok ügyében is segíteni tudunk. Az LME már korábban is támogatott terveket, azonban hatékonyabb és szerteágazóbb támogatási rendszerre van igény.

Sz. Gy.: *Terveztek-e szakembereket, nyelvészeket bevonni a munkába?*

F. T.: Igen, ez fontos pont. Sajnos egyelőre nem rendelkezünk olyan anyagi háttérrel, hogy fizetett nyelvészeket alkalmazzunk, ezért nehéz a kérdés. Reméljük, hogy találunk olyan szakembereket, akik segítenek nekünk. Mindenképpen fontos pont még egy olyan terület (levelezőlista, webes felület) kialakítása, ahol a felmerült és már megválaszolt kérdéseket, valamint a nyílt keretek között elkészített fordításokat tesszük közkinccsé.

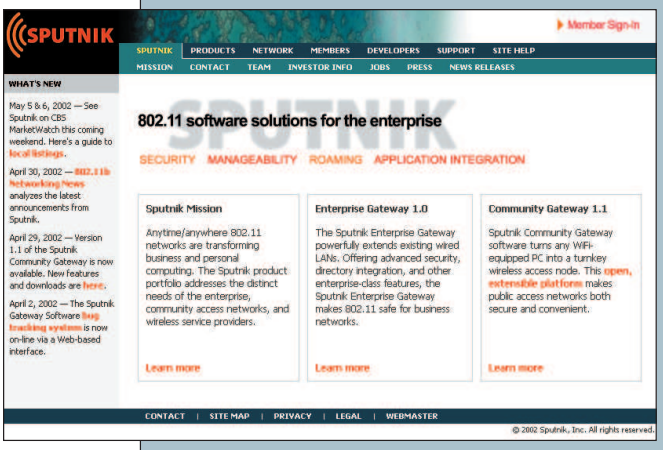
Az Egyesület belső munkájában is szervesen részt kívánunk venni, gondolok itt rendezvényekre, az oktatással foglalkozó emberek segítésére, illetve a marketinggel vagy a CD-író részleggel történő együttműködésre. Ide tartozik még egy fontos dolog: ellenőrzőfelületet szeretnénk biztosítani a folyamatos anyagok (például az LME hírlevél vagy a kisebb leírások) számára is.

Sz. Gy.: *Köszönöm a beszélgetést!*

Szy György, a Linuxvilág főszerkesztője

Sputnik Wireless Network a Linuxcare alapítóitól

Miközben mindenki a Boingóra figyelt (ez az új nemzeti, vezeték nélküli internetrendszer, melynek feje *Sky Dayton*, az EarthLink alapítója), a Linuxcare három alapítótagja – *Dave Sifry*, *Dave LaDuke* és *Art Tyde* – csendben felépítette saját osztott modellen alapuló rendszerét, amelynek a szabad programok, a nyílt forrás és a Linux-mozgalom nyitott utat.



A vállalat neve Sputnik, és az óvatos kezdet után végre útjára indították a szolgáltatást.
Ahhoz, hogy a Sputnikot el tudjuk helyezni, a következőképpen csoportosítsuk a vezeték nélküli hálózatokat (amilyen például a 802.11b vagy a WiFi):

1. zárt, vezeték nélküli ethernet helyi hálózat,
2. a Világháló elérése vezeték nélkül, díj ellenében,
3. a Világháló elérése vezeték nélkül és ingyen.

A fenti három bármelyike megjelenhet a vezeték nélküli ügyfélprogramban, ha a hordozható számítógép valamelyik „forrópont" hatósugarába kerül. De csak az utolsó két típussal juthatunk ki a Hálóra. A Boingo és a Sputnik közti leglényegesebb különbség az, hogy a Boingo a kettes típus szerint köti össze a vállalkozásokat, míg a Sputnik a kettesre és a hármasra is kiterjed – de főleg a hármasra. Felfoghatjuk úgy is, mint egy díj fejében használható, értékes bővítményt, amely világszerte valamennyi résztvevő forrópont számára megfelelő.
A különbséget a kapcsolatok tükrében is vizsgálhatjuk: a Boingo a PC-k világából való, a Sputnik pedig a Linux/Unix-világból. A Boingo ügyfélprogramját és szolgáltatásait díj ellenében bocsátja a felhasználók rendelkezésére, ugyanakkor lehetővé teszi, hogy ezekkel a felhasználókkal a szolgáltatók egységes módon lépjenek kapcsolatba, egyenes bevételi forrással. A Sputniknál a felhasználó nemcsak élvezi a WiFi sávszélesség előnyeit, hanem tovább is adja.
A Linuxnak megfelelően a hordozható gép egyszerre ügyfél és kiszolgáló. A résztvevő a rendszer teljes jogú részévé válik.
A Sputnik Gateway Software használatával továbbadod a sávszélességet, miközben magad is használod.
A géped elérési ponttá, úgynevezett forróponttá válik. A Sputnik átjárója azonban nem általános elosztóként, hanem okos útválasztóként működik, és a forgalmat a többi Sputnik-tag felé továbbítja. A rendszer minden új taggal tovább bővül. Dave Sifryt idézve:
„Az átjáró egy programozható bővítőeszköz, melynek használatához nincs szükség különleges ügyfélre. Semmilyen programot nem kell letölteni a gépre! Teljes mértékben szabványok szerint működik."
Glenn Fleishmann (802.11b Networking News) véleménye szerint:
„Egyszerűen lenyűgöző! A sputnikosok minden gondot megoldottak, amit a Boingónak nem sikerült, miközben érdekes, vírusként viselkedő megoldást kínálnak. Megoldották a tűzfal (a helyi hálózatok védettek, míg az AP nyílt), a hitelesítés (kötött portál minden munka nélkül) és az elsőbbség kérdését (a helyi felhasználók és a Sputnik-tagok másokkal szemben elsőbbséget élveznek). A taggá válás a tag számára a hálózat többi részét ingyenesen nyitja meg, ami két olyan szempontból is fontos szerepet játszik, amelyek az Internet kapcsán általában felmerülnek, és a vezeték nélküli, közösségi hálózatok esetén pedig különösen. Az egyik a jól felfogott önérdek, mivel az egyén hozzájárulása növeli a hálózat méretét és annak valószínűségét, hogy mások is csatlakoznak; a másik a nagylelkű önzetlenség, mivel az egyén tulajdonképpen semmit sem nyer azáltal, hogy saját elérési pontja és hálózata használatát másoknak lehetővé teszi. Csak a két tulajdonság kombinációja eredményez vírusszerű viselkedést.
A most következő megállapítással természetesen lehet vitatkozni: ez a megoldás a közösségi hálózatot választja egy előre gyártott, PC-alapú csomag révén, amely zárt dobozhoz hasonlít. Pénzt csak a kívülállóktól kapnak, akik nem elég nagylelkűek vagy tapasztaltak ahhoz, hogy társakká váljanak.
Kíváncsi vagyok, hogy a fenti megjegyzés milyen kritikát fog kiváltani.
Az átjáró tűzfallal van ellátva, ezen túlmenően alkalmazási felületként működik, gyorstárazást alkalmaz, nyomon követi a használatot, kezeli a hitelesítést, a távoli irányítást és egyéb dolgokat. Az alaptechnológiák ugyancsak nyílt forrásúak, és GPL alatt érhetők el.
Az előfizetők havi díjat fizetnek, de korlátozott ideig (legalábbis lapzártánk idején) ingyenesen használhatják a hálózatot. A Sputnik honlapja:
➲ http://www.sputnik.com.

Linux Journal, május 97. szám

Doc Searls
(doc@ssc.com) a Linux Journal szerkesztője, havi rovata a Töprengő, valamint a Cluetrain Manifesto társszerzője.

Egyértelmű?

Kérdések

1. Mely személy weboldalának a címe a következő: „Egy világháló-analfabéta honlapja”?
2. *Linus Torvalds* a következőket jegyzi meg egy bizonyos személy nyílt forrású alkalmazásokról szóló beszédével kapcsolatban: „Inkább hallgatom Isaac Newtont, mint X-et. Igaz, hogy már háromszáz éve halott, de kevésbé bűzlik tőle a terem.” A kérdés egyszerű: kiről beszélt Linus?
3. Mi a pingvinek két csoportjának gyűjtőneve?
4. *Vinod Valloppillil* (Microsoft) néhány hízelgő kijelentést tett a Linuxszal kapcsolatban:
   „A Linux a Unix-rendszerek legkiválóbb képviselője, amelyet nagyon fontos alkalmazások futtatásához használnak.”
   „A Linux alkalmazása a fontos feladatokat ellátó rendszerek esetében nagy elismertségnek örvend.”
   „Összehasonlításképpen: ugyanazon a gépen korábban IE4-NT4-rendszert, majd egy Linux-Netscape Navigator-együttest futtattam, az utóbbi 30–40 százalékos sebességnövekedést mutatott a HTML és a grafika megjelenítésében.”
   Honnan származnak Vinod Valloppillil hízelgő megjegyzései?
5. Köztudott, hogy Linus a Helsinki Egyetem Számítástudományi Intézetében tanult, amikor elkezdte a Linux fejlesztését. Mi Linus anyanyelve?
6. *Ray Tomlinson*, aki Cambridge-i BBN-nél dolgozik kutatóként, valami különlegesen fontosat hajtott végre 1971-ben. Mi volt ez?
7. Egy etimológiai kérdés: *William Gibson*, a Neurománc című ismert regényében egy új szót alkotott. Mi ez a szó?
8. Mi Linus Torvalds második keresztneve?
9. Melyik alkalmazásról szólnak a Linux Journal következő sorai: „Ha az emberek már igeként használják programod nevét, biztos lehetsz a sikerében”? Ugyanebben a cikkben a következőket is olvashatjuk: „Nem lehet véletlen egybeesés, hogy e program terjedésével a Linux lekerült arról a listáról, amely a könnyen feltörhető rendszereket tartalmazza.”
10. Egy jelentős mű szerzője a köszönetnyilvánítást a következő sorral fejezi be: „… és köszönet az AT&T Bell Labsnek, hogy kirúgtak, és ezáltal megszülethetett a mű.” A mű jellemzésére vonatkozóan a Wired című magazin a következőket írja: „Egy könyv, amely kiadásának a Nemzetbiztonsági Hivatal örült a legkevésbé.” Ezek alapján a kérdés a következő: mi a könyv címe és ki az írója?

Válaszok

1. Linus Torvalds
   ➲ http://www.cs.helsinki.fi/~torvalds
2. *Craig Mundie*-ről, a Microsoft alelnökéről.
3. Totyogó pingvinek vagy úszó pingvinek. A vízben levő pingvinek csoportját úszó pingvineknek nevezzük, a jégen járó pingvinek csoportját pedig totyogó pingvineknek. Ezen megfogalmazások a Negyedik Nemzetközi Pingvin Konferencia (Chile, 2000) munkáját dicsérik.
4. A híres-hírhedt Halloween-dokumentumokból.
   Aki nem hallott a Halloween-dokumentumokról, a következő címen megtalálhatja őket:
   ➲ http://www.opensource.org/halloween/halloween1.html
5. Igaz,hogy Linus Helsinkiben, Finnországi fővárosában nőtt fel, de az anyanyelve svéd. Finnországban jelentős a svéd anyanyelvű lakosság, akik magukat finlandssvenskeknek nevezik.
6. Ő küldte a világ első e-mailjét hálózaton keresztül. Véleménye szerint ez olyan ártalmatlan volt, mint a QUERTYIOP.
7. Kibertér (cyberspace).
8. Benedict.
9. *Fyodor* Nmap nevű programja. A cikk a szerkesztők által legjobbnak ítélt programokat írja le, az Nmapet a legjobb biztonságtechnikai eszköznek választották.
10. *Bruce Schneier*: Alkalmazott kriptográfia.

Sumit Dhar

SuSE javító CD

A Linuxvilág olvasói abban a szerencsés helyzetben vannak, hogy augusztusi számunk egyik CD-mellékletén a SuSE Linux 8.0-hoz addig megjelent összes hibajavítást megtalálhatják. Ennek a korongnak és a YaST2-nek a segítségével rendszerüket pár gombnyomással frissíthetik. Ezt azért is fontos megtenni, mivel a fejlesztők nagyon sok hibát kijavítottak a megjelenés óta.

Mivel a YaST2 csomagtelepítéshez mind rendszergazdai jogokra, mind a grafikus felülethez való hozzáférésre szükség van, a rendszergazda módba történő váltáshoz a `sux -` parancsot kell használni a `su -` helyett. Ezután indítva a YaST2 barátságos és kényelmes felületén frissíthetjük rendszerünket.

Csontos Gyula

Piaci pillanatkép

Beszélgetés dr. Szentiványi Gáborral, a SuSE Linux AG magyarországi irodájának igazgatójával.

A teljes körű Linux-alapú megoldásokat kínáló SuSE hazai képviselete másfél évvel ezelőtt nyílt meg. Elsősorban a nünbergi székhelyű cég termékeit forgalmazzák, emellett fejlesztéssel és honosítással is foglalkoznak.

A felhasználók számára szolgáltatások széles körét nyújtják, mint amilyen a SuSE Linux tanfolyamok szervezése, illetve a szakértői tanácsadás.

Dr. Szentiványi Gábor igazgatót az iroda eddigi történetéről kérdeztük, elsősorban az általuk végzett fordítási munkákról, valamint a United-Linux létrejöttének hatásairól. A UnitedLinux létrehozására, mint ismeretes, a négy vezető Linux-nagyvállalat kötött szövetséget, melyek egyike a SuSE. A májusban bejelentett kezdeményezés elsődleges célja egy egységes, vállalati alkalmazásra készülő Linux-változat kifejlesztése, és ezáltal a Linux-használat vállalati környezetben történő felgyorsítása. Az együttműködésnek köszönhetően az alapító cégek – a Caldera, a Conectiva, a SuSE és a Turbolinux – hamarosan az egész világon képesek lesznek terméktámogatás és minden egyéb kapcsolódó szolgáltatás nyújtására. A UnitedLinux a magyar nyelvű telepítést is lehetővé teszi majd.

Szabó Ágnes: *Hogyan indult az iroda tevékenysége?*

Szentiványi Gábor: A SuSE hazai képviseletével egyedül kezdtem el foglalkozni egy-másfél évvel ezelőtt. Jelenleg hatan dolgozunk a cégnél, emellett vannak külső segítőink – itt kiemelném az LME szerepét –, akik a programcsomagok és a könyvek fordításában működnek közre.

Sz. Á.: *Felvetődhet a kérdés az iroda működése kapcsán, hogyan fér össze a Linux gondolatisága és a gazdasági tevékenység?*

Sz. G.: Alapvető célkitűzésünk, hogy minél több minden bekerüljön a köztudatba a Linux területén. Egyrészt igyekszünk terjeszteni a SuSE termékeit, másrészt pénzügyileg is működőképesnek kell lennünk. A linuxos társadalmat valóban megosztja, ha valaki úgymond pénzt csinál mindebből. Egyelőre ez az irány hazánkban még új, ezért sokan nem fogadják el.

Sz. Á.: *Hogyan kerülnek hozzátok a SuSE fejlesztései?*

Sz. G.: A SuSE fejlesztői a kezdetekkor lefektettek egy automatizált folyamatot, az úgynevezett Autobild rendszert. A terjesztés (distribution) fejlesztési anyagához nekünk is hozzáférésünk van. Az Autobild segítségével egységesen készítjük minden felületre az operációs rendszereket, ezáltal a programok azonos szintű minőségét tudjuk biztosítani. E rendszernek legfőbb előnye, hogy nem sok PC kategóriás gépen fut, így nincs szükség különleges gépekre, és a rendszer rugalmasan méretezhető. Amellett, hogy központi a fejlesztés, adott a lehetőség nemzeti nyelvek beépítésére is, a SuSE Linux 8.0 operációs környezetnek például magyarítani tudtuk a libc-jét, mivel a libc-forrásban hozzá tudunk férni a csomaghoz.

Sz. Á.: *A magyarításhoz kapcsolódva, úgy látod, érdemes volt ezzel foglalkozni?*

Sz. G.: Igen, hiszen ma már a magyar változatok készítése elkerülhetetlen. Az emberek ugyanis el vannak kényeztetve. Akár csukott szemmel is bekapcsolhatják a számítógépet, elkezd futni rajta a Windows, és örülnek, milyen kezes. Figyelembe kell venni mind a kezdő felhasználók, mind a gyakorlott (poweruser) igényeit. A kezdő felhasználók számára a termék kezelésének egyszerűségét, a nagyobb tudású felhasználóknak hatékonyságot, a profiknak pedig rugalmasságot kell kínálnunk. A magyarítás pedig biztosítja mind az egyszerűséget, mind a hatékonyságot. Amikor egy körkérdéssel felmértük a felhasználók elvárásait, az derült ki, a legfontosabbnak a magyar változatot és a leírást tartják. Mi ez utóbbit is nyújtjuk: nyomtatott formában közel kilencszáz oldal leírás érhető el. A fordítás sikere egyébként jelentősen meglátszott az eladási adatokon is.

Sz. Á.: *A UnitedLinuxban milyen szerepet játszik a SuSE?*

Sz. G.: A UnitedLinux valójában nem termék, hanem termékalap. A közös alapra mind a négy cég felépítheti majd a saját termékét. Az új fejlesztéseken a United-Linux cégjelzése is szerepel majd, azonban mind a négy alapító megtartja a saját márkanevét. A UnitedLinux elsősorban a kiszolgálóvonalon fog futni, asztali gépekre nem szeretnénk bevinni. A SuSE az együttműködés folyamatának teljes technikai felügyeletét felvállalta, folyamatos kapcsolatban áll a fejlesztőkkel. A United-Linux létrehozása egyébként nagyrészt a piacfelosztásról szól. A linuxos vállalatoknak még nincs olyan szintű, egész világra kiterjedő tevékenysége mint például az IBM-nek. A négy vállalat a UnitedLinux segítségével más és más területekre összpontosít. A Caldera elsősorban Amerikában, a Conectiva Dél-Amerikában erős, a Turbolinux Ázsiára összpontosít, a SuSE pedig főként Európára. Voltak már viták, a Caldera ugyanis munkaállomásonkénti (per-seat) engedélyezésben (felhasználási szerződésben) gondolkodik, azonban a SuSE ebbe a maga részéről nem egyezett bele.

Sz. Á.: *Ez elég nagy visszhangot keltett linuxos körökben. Ezek szerint ezt a megoldást csak a Caldera szeretné?*

Sz. G.: Igen, a SuSE már hivatalosan is bejelentette, hogy semmiképpen nem lesz gépenkénti felhasználási szerződése (license).

Sz. Á.: *Mekkora a SuSE hazai tábora?*

Sz. G.: Véleményem szerint elég nagy tábora van. A SuSE felhasználói táborában egyaránt megtalálhatóak a kezdő és haladó felhasználók, és viszonylag szolidnak mondható, más közösségekkel ellentétben nem rántanak kardot, ha valaki más körbe tartozik. Nem radikálisak, egyszerűen használják a rendszert. Hogy mennyien vannak, azt nem tudjuk. Számos felhasználó jegyezte be magát terméktá-

mogatási oldalunkon, mivel a kilencven napos segítség a bejelentkezéshez kötött. De mivel sokan ellenérzésekkel viseltetnek a bejegyzésekkel szemben, elég nagy lehet azoknak a köre, akikkel mi nem találkozunk.

Sz. Á.: *Eljutnak-e a SuSE termékei a hazai közintézményekbe: iskolákba, önkormányzatokba?*

Sz. G.: Mint az közismert, a Széchenyi-terv kapcsán lezajlott az SZT-IS-3 és az SZT-IS-3 B pályázat. Az első alkalommal mintegy tizenháromezer pedagógus juthatott számítógéphez, azzal a kikötéssel, hogy mind Windows, mind Linux legyen a gépen. Gondok igazából a kivitelezéssel voltak. A pályázatban javasolták többek között a SuSE-t is: bár úgy volt megadva, hogy ez egyfajta ingyenes lehetőség. Ez alapvetően igaz is, de a termék előállítása nyilván pénzbe kerül. Az ingyenesség miatt senki sem számított költségekre a telepítés kapcsán. A beszállítók megjelentek nálunk, és kérték a tizenháromezer SuSE-csomagot. Végül CD-ket adtunk át nekik, hogy azt másolják le. Mint később kiderült, a pedagógusok hiányolták a kézikönyvet, tehát az egész csomagot, amit a Microsofttól viszont megkaptak. A második alkalommal az SZT-IS-3 B pályázat kétezer gépre szólt. Ekkor már úgy döntöttünk, megelőzzük a gondokat, és mi magunk gyártjuk le a CD-ket. Bár mindez nem a mi feladatunk lett volna, nekünk kellett foglalkoznunk vele. A pedagógusok ismét azzal kerestek meg, hogy ugyan a CD eljutott hozzájuk, de a könyvet hiányolták, egyébként érthető módon. Az eredmény, hogy jelenleg a tanárok 15 százaléka kapta meg a CD-t. Folynak a tárgyalások, hogy a teljes SuSE csomag minden érintett pedagógushoz eljusson.

Sz. Á.: *Mik a terveitek, hová kívántok fejlődni?*

Sz. G.: A fejlődés útja szerintem a szolgáltatásokban rejlik. A termékek, amelyeket forgalmazunk, két vonalon futnak. A magánfelhasználóknak, kisvállalkozásoknak szóló termék mellett vállalati kiszolgálótermékeket is kínálunk. Az üzleti élet igényeinek megfelelő Linux-környezetet nyújt a SuSE Linux Enterprise Server. A két változat minősége hasonló, azonban a vállalati felhasználásra szánt változatban más a cél. Ez növelt megbízhatóságú rendszer, minősített alkatrész-támogatással: nem változik gyorsan, nem „mozgó célpont". Hibák esetében csupán a szükséges javításokat végezzük el, mely a rendszer stabilitását nem veszélyezteti. Úgy gondoljuk, hogy ez az üzleti modell megkerülhetetlenné, alapvetővé fog válni a Linux világában. A karbantartás futamideje legalább két év, melyből 12 hónapot a csomag ára tartalmaz, ez meghosszabbítható. A SuSE Linux Enterprise Servert az egyik legnagyobb hazai vállalat is megvette tőlünk IBM típusú gépeihez. A SuSE Linux Enterprise Serverben egyedülálló módon kiválasztható a magyar nyelv, így mindenki számára könnyen telepíthető.

Sz. Á.: *Milyen lépéseket tartanál fontosnak, és mit tud tenni a SuSE hazai képviselete, hogy a Linuxot használók köre tovább bővüljön Magyarországon?*

Sz. G.: Amire szükség lenne, hogy linuxos alapon legyenek oktató-, könyvelő-, bérszámfejtő, szórakoztató és más hasonló programok, és ami nagyon lényeges, magyar nyelven. A hazai alkalmazások fejlesztését tartom fontosnak. Bár mi nem alkalmazásfejlesztő, hanem alaptechnológiát kínáló cég vagyunk, mi is próbálunk lépéseket tenni ennek érdekében. Közelebb kellene vinni az emberekhez a Linuxot, és ebben sokat segítene, ha minél több magyar nyelvű nyílt forrású alkalmazást kapnának a felhasználók. Fel kell térképezni, hogy a Linuxban hol vannak hiányosságok, és ott meg kell adni a szükséges támogatást. Ami még szerintem nagyon lényeges, hogy a Linuxnak egységes hátteret kell biztosítani. A piacon ugyanis úgy képes megmaradni, hogy bár valójában változatos és sokszínű, kifelé egységességet tud felmutatni. A Linux nagy előnye, a változatosság piaci szempontból sajnos hátrány is lehet. A vásárlóközönség csak akkor bízik meg termékeinkben, ha látja, hogy megbízható háttér, komoly vállalat áll mögöttünk. Ezt ma már nyújtani kell a Linux területén is. Ez a SuSE esetében megvan: a világon a SuSE-nek van a legnagyobb fejlesztő csapata, mintegy háromszázan dolgoznak célirányosan nekünk. Minden fejlesztés bekerül a kiadásokba, és nyílt forrású lesz. Egymás mellett kell léteznie tehát a nyílt forrásnak, és azoknak is, akik mindezt eladják. A nyílt forrás alapvető fontosságú: a SuSE szinte az egyetlen, ami nem tesz olyan meghajtókat a kiadásaiba, amelyek nem nyílt forrásúak (nVidia, számos winmondem). A YaST viszont nem vehető ki a terjesztésből, ez teszi a SuSE-t SuSE-vé, annak része, mivel nem akarjuk, hogy a SuSE hozzájárulása nélkül valaki anyagi haszonhoz jusson a termékből. Mindenki módosíthatja, másolhatja, közzéteheti, csakhogy ebben az esetben nem használhatja a SuSE cégjelzését.

Sz. Á.: *A kézikönyvek vagy azok részletei is másolhatók?*

Sz. G.: Igen, a szerzői jog jelzésével (copyright), amint ez szerepel is a könyv első lapjain. Ha valaki csak oldalakat másol, akkor a lap fejlécébe be kell tennie a szerzői jogot. A célunk az, hogy közkinccsé tegyük a SuSE fejlesztéseit de a dokumentációnál a SuSE név feltüntetéséhez és az ingyenes terjesztéshez ragaszkodunk. Ennek érdekében hivatalos helyeken lobbizást is igyekszünk folytatni, ezt azonban a Linux filozófiájával sokszor nehéz összeegyeztetni. A lobbizásba beletartozik a politika is, azonban az elveket nem szabad feladni. De ha sikerül minél szélesebb körben elterjeszteni, az a Linuxnak is jó.

Szabó Ágnes
(agi@netelligents.com) újságíró, szociológus. 25 éves, sajtó-, internet- és kávéfüggő. Ez utóbbiról szerencsére leszokófélben van. Kedvenc sportja a futás és a lovaglás.

Nézzük, mi is az a CVS!

A CVS (Concurrent Versions System) rendkívül összetett változatkövető és kezelőrendszer. A fejlesztés közben keletkező forráskódok rendezésére, változásainak követésére szolgál, és arra, hogy mindezt bármely felhasználó, illetve az arra jogosultak számára elérhetővé tegye.

Először is nézzük a CVS előnyeit és hátrányait!

Előnyei:

- A kód jogosulatlan felülírásának kiküszöbölése.
  Ez az előny elsősorban akkor tekinthető valódi előnynek, ha a fejlesztők száma nagy, hiszen egy kisebb fejlesztőcsapatban ez a gond még megoldható. Ezen túlmenően akkor sem nagy nyereség, ha a forráskód maga annyira moduláris, hogy több fejlesztő biztosan nem nyúl ugyanahhoz az állományhoz.
- Változatkövetés.
  A változatkövetés nagyobb projektek fejlesztésekor szinte mindig jókora segítség, hiszen így nemcsak az emberi tényezőt kell figyelembe venni, de még a gép is segít nekünk abban, hogy az egyes részkódok a megfelelő számozást kapják, így nyomon követésük és a fejlesztés dokumentálása nagymértékben könnyebbé válik.
- Naplózás a készítés lépéseiről.
  A CVS naplózza, ha valaki hozzányúl a kódhoz, de mi magunk is képesek vagyunk a naplóhoz megjegyzéseket fűzni, így újfent csak a fejlesztés nyomon követése és dokumentálása lesz egyszerűbb.
- Kódváltozás figyelése.
  A CVS figyeli, ha egy kódrészlet megváltozik, jelzi, amikor egy forráskódról adatot kérünk, sőt, ezt a változást a megfelelő formátumban akár le is kérhetjük a CVS-ről, így egy egyszerű `patch` nevű programmal meglévő forráskódjainkat kézzel is javíthatjuk.
- Frissesség.
  A CVS használatából adódóan mindig naprakész, friss kódokat tartalmaz, bár ezek a legtöbb CVS-kiszolgáló esetében fejlesztői változatok, ám a legújabb forráskódot ettől függetlenül mindig CVS-ből tölthetjük le.
- Biztonságos azonosítás.
  A CVS képes azonosítani és különböző jogokkal felruházni a felhasználókat. A CVS többféle azonosítási és használati lehetőséget biztosít, ezekről azonban a későbbiek folyamán ejtünk szót.

Hátrányai:

- Viszonylag nehézkes kezelhetőség.
  Maga a CVS rendkívül összetett feladata és biztonsági eljárásai miatt nem tartozik a könnyen kezelhető programok közé. Néha még a grafikus alkalmazások használata is nehézkes és időrabló (a gcvs, xemacs beépített CVS kezelője stb.).
- Támadások elleni alacsony megbízhatóság.
  Azoknál a fejlesztőcégeknél, ahol a forráskód titkos vagy nem adható ki, az Internetre kitett CVS nem nyújt megfelelő biztonságot, hiszen nekik nagyon nem ajánlott a CVS ilyen használata. Mindezek mellett a CVS nem az a könnyen feltörhető típus.
- Lassúság.
  A CVS, ha többen használják, lelassulhat. Igaz, ez a jelenlegi internetsebesség mellett nem annyira feltűnő, de egy belső hálózaton már meglátszik, hogy a CVS „gondolkodik".

Hol érdemes használni a CVS-t? Érdemes-e használni?

Természetesen a válasz határozott *igen*, hiszen a CVS nagyszerű eszköz a fejlesztők számára. A közös kódot nem szükséges valakinek minden órában összefésülnie, nem kell vigyáznunk arra, nehogy felülírjuk a másik munkáját, mert a CVS ügyel rá. A CVS-t leginkább nyílt forrású projektek fejlesztésekor érdemes használni, hiszen a fejlesztők nagy száma mellé itt rendkívül kevés koordinátor társul, akiknek munkáját egy CVS-rendszer nagymértékben megkönnyíti. Természetesen nagycégeknél is érdemes meggondolni a CVS használatát, mert ha a belső CVS-t elzárják a külvilágtól, a belső fejlesztőcsapat munkája jelentős mértékben hatékonyabbá és gyorsabbá válik. Mindenesetre nem árt némi felkészülés, mielőtt egy fejlesztőcég bevezeti a CVS-t, mert a kezelése tényleg nem túl egyszerű, használata pedig néha nehézkes, a munka azonban néhány jó héjprogrammal megkönnyíthető.
Kezdjük meg hát az ismerkedést a CVS-sel!

Honnan szerezhetünk CVS-t?

A legegyszerűbb, ha letöltjük a CVS honlapjáról
➲ http://www.cvshome.org.
Ezenkívül a CVS csomag bármely Linux-változatban megtalálható. Ha beszereztük, telepítsük:

```
apt-get install cvs
rpm -i cvs
```

A telepítés folyamán néhány kérdésre kell válaszolnunk, elsőként a `repository` helyéről érdeklődik:

```
Where are your repositories?
```

Alapértelmezetten a */var/lib/cvs* van megadva, amely nem rossz döntés, ám ha más elképzelésünk van arról, hogy a CVS hol tárolja azokat a forráskódokat, amelyeket a fejlesztők töltenek fel, adjunk meg neki más helyet. Arra azért ügyeljük, hogy lehetőleg minél több helyet biztosítsunk a CVS számára, mivel egy projekt akár egy pillanat alatt is hihetetlen méreteket érhet el.
Másodikként a *history* fájl forgatására kérdez rá:

```
Do you want the history files in your
➥repositories rotated weekly?
```

Ha azt szeretnénk, hogy history fájlunkat a rendszer hetente forgassa, válaszoljunk igennel, egyébként nemmel. Itt választhatjuk az `individual` lehetőséget is. Ha igennel válaszolunk, akkor a telepítő megkérdezi milyen időközönként szeretnénk forgatni a helyét. A forgatáshoz a `cron` vagy `anacron` démonra van szükségünk.
A következő kérdés a `pserver` engedélyezésére vonatkozik:

`Should the CVS pserver be enabled?`
Ez a CVS alapértelmezett kiszolgálótípusa.
Erről a későbbiekben még szó esik, de ha hálózaton keresztül szeretnénk elérni a CVS-t, és nincs SSH-nk, válaszoljunk igennel.
Ezután a CVS-kiszolgáló már fent is van. A pserver `inetd`-ből fut, így ha engedélyeztük a `pservert`, érdemes elindítani. A `pserver` a `2401`-es kapun várakozik a kapcsolódásra.

A CVS általános változói:

Ezek környezeti változók, rendszerünktől függően az alábbi formátumok valamelyikét kell használnunk:

- `export VALTOZO="ØrtØk";`
- `setenv VALTOZO "ØrtØk"`
- `SET VALTOZO="ØrtØk"  (DOS alatt).`

A `CVSROOT` a számunkra érvényes CVS-főkönyvtár beállítása. A példában belátható CVS kiszolgáló üzemmódban működik.

```
CVSROOT=":pserver:cvsuser@cvshost.hu:
➥/CVS/root/k nyvtÆr"
```

Amikor a CVS kiszolgálóként dolgozik, más protokollokat is használhatunk (ebben az esetben a `CVS_SERVER` változóban adhatjuk meg a használni kívánt protokollt):

```
CVSROOT=":ext:cvsuser@cvshost.hu:
➥/CVS/root/k nyvtÆr"
```

Ekkor a CVS nem működik kiszolgáló üzemmódban.

```
CVSROOT="/CVS/root/k nyvtÆr"/
```

Még sokféle értéket adhatunk a CVSROOT-nak, erre később térünk vissza.

```
CVSROOT=":met dus:felhasznÆl nØv@
➥gØpnØv:repository_k nyvtÆra"
```

CVS_CLIENT_LOG

A naplófájl neve és teljes elérési útja. Lehetőség nyílik egy olyan fájlnév megadására, amelybe a CVS azt naplózza, hogy az ügyfél éppen milyen műveletet hajtott végre, és arra milyen választ kapott. A naplófájl alapján a CVS létrehoz egy *$CVS_CLIENT_LOG.in* és egy *$CVS_CLIENT_LOG.out* fájlt. Értelemszerűen a *.in*-ben a kimenő üzenetek az ügyfél által küldöttek, míg a *.out*-ban a kiszolgálótól kapott üzenetek találhatók meg. Természetesen ezt a naplózást a CVS csak akkor használja, ha kiszolgáló üzemmódban használjuk.

CVSIGNORE

Azon fájlok listája, amelyek a CVS számára nem láthatók. A fájlokat szóközökkel kell felsorolni.

```
CVSIGNORE="elso.txt
➥masnaknem.c proba.c proba.h"
```

CVSREAD

Ha ennek a változónak értéke van, a CVS a letöltött fájlokat csak olvashatóra állítja, amelyeket így nem tudunk módosítani, amíg írható-olvashatóra át nem állítjuk őket.

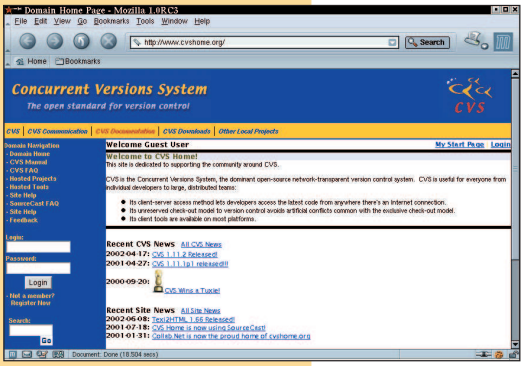
CVSUMASK

Ezzel a változóval tudjuk beállítani a CVS-be került fájlok hozzáférési jogát.

EDITOR | CVSEDITOR

Megadhatjuk kedvenc szövegszerkesztőnket.
A `cvseditor` felülbírálja az `editor` változó beállítását. A szerkesztőt a megjegyzések naplóba írásához használjuk (lásd később). Ha egyik változónak sem adtunk értéket, az alapértelmezett szövegszerkesztőt használjuk, ami legtöbb esetben a `vi` vagy a `nano`, esetleg az `ae`.

CVS_PASSFILE



Ebben a változóban írhatjuk felül a CVS jelszófájljának alapértelmezett helyét: *$HOME/.cvspass*. Itt tárolja a CVS a különböző `CVSROOT`-okhoz adott jelszavakat.

CVS_SERVER

Ebben a változóban tudjuk a CVS-ügyfélnek megadni, hogy a kiszolgáló, amit el szeretnénk érni, milyen eljárást használ a hálózati kapcsolattartásra. Alapértelmezettként a `cvs` érték van megadva, amit érdemes `ssh`-ra módosítani – így az ügyfél is a jól ismert SSH-protokollt fogja használni. Természetesen ehhez az ügyféloldalon az SSH-ügyfélre (`ssh`), a kiszolgálóoldalon pedig az SSH-kiszolgálóra (`sshd`) van szükség.
A CVS változóit héjunk *.rc* fájljában érdemes beállítani, hiszen ha sokat dolgozunk CVS-sel, kényelmetlen lenne mindig újra és újra beállítani. Mindenképpen figyeljünk ezekre a változókra, mivel rengeteg felesleges munkát és időpocsékolást takaríthatunk meg.
Maga a CVS arra szolgál, hogy egy nagyobb fejlesztőcsapat munkáját összehangolja, figyeljen a különböző állapotok, szálak változataira, forráskódjára. A CVS segíthet nekünk a kód rendszerezésében, a leírás elkészítésében és a zökkenőmentes munka összehangolásában.
Néhány program, amit CVS-ben fejlesztenek:

- PHP, a közkedvelt parancsnyelv,
- MPlayer, a linuxos filmlejátszó,
- Wine,
- gcc, a GNU C fordító,
- CVS, a változatkövető rendszer.

A CVS előnyeivel és hátrányaival együtt a legjobb és legmegbízhatóbb változatkövető rendszer, így használata melegen ajánlott – legyen rendszerünk akár Linux, akár BSD, netán DOS.

Vass Endre
PHP-programozóként dolgozik, szabadidejében éppen egy számítógépes szerepjátékot fejleszt C/CC++ nyelven. Szereti a régi számítógépeket, kedvenc változata a Debian GNU/Linux.

Mese két bazárról: a piacról és a rendszermaglistáról

1999 augusztusában a Linux World Expón találkoztam az IBM néhány képviselőjével, és megtudtam tőlük, hogy a vállalat gyanakvó érdeklődéssel kíséri ezt az operációs rendszert. Elmondták, hogy a Linux teljesen váratlanul jelent meg náluk, és (minden stratégiai megfontolás nélkül) egyre több kiszolgálón tűnt fel szerte a cégnél. Így aztán az IBM felmérést végzett az egyik részlegen belül, amelynek célja a Linux ismertségének feltérképezése volt. A skála egyik végén az „elboldogulok a Linuxszal", a másik végén a „rendszerkódot gyártogatok" képesség szerepelt.
Az eredmény: mind a 600 megkérdezett személy elboldogult a Linuxszal, 120 pedig képes volt rendszerkódot gyártani. Ez a felfedezés kétségkívül a vállalat stratégiai elkötelezettségét jelezte a Linux operációs rendszer iránt.
Azóta eltelt két és fél év, és a Linux jelentős operációs rendszerré vált. Ez volt a hírösszefoglaló a legutóbbi Linux World Expóról, amely – amikor ezeket a sorokat írom – éppen csak véget ért.
Az IBM felvonultatta jelentős ügyfeleit: *L. L. Bean*-t, a Boscovt, a Pixart és *Solomon Smith Barney*-t.
A Hewlett-Packard a Dreamworks SKG-t mutatta be, az Egenera pedig a Credit Suisse First Bostonnal kötött egyezségéről számolt be.
Az LWE nyitó felszólalásában *Carly Fiorina* (HP) azt nyilatkozta, hogy 2002 a Linux „kitörési" éve lesz, utalva arra, hogy a Gartner elemzése a gazdasági hanyatlás ellenére 15 százalékos növekedésre számít. A HP néhány Linux-terméket is bejelentett a vállalati és a telekommunikációs ügyfelek számára, és referenciaként az Amazon, a BMW, a Boeing, a Speedera, a ViaWest és a Verizon nevét említette.
Ugyancsak az LWE-n *Holger Dyroff* (SuSE) ezt mondta: „Egészen a közelmúltig a szakmabeliek választották a Linuxot. Mostanra azonban éppen az ellenkezője igaz." Vezető informatikai tisztviselőktől érkeznek megrendelések a SuSE támogatási szerződésére IBM nagygépekhez. Hetente 2–3 ilyen vásárlási megrendelés is érkezik, szerződésenként 4 500–11 500 dollár értékben.
„A közel másfél éve kapható nagygépes változatot választó ügyfelek egyharmada bank" – mondta Dyroff.
Az IBM különleges akciót hirdetett meg zászlóshajója, a zSeries (ismertebb nevén S/390-es) nagygépek csak linuxos változatára.
Sőt az IBM, amely a múlt évben nagy hangon hirdette, hogy egymilliárd dollárt költ a Linuxra, most azt állítja, hogy a befektetés nagy része már megtérült. Aligha ők az egyetlenek, akik az eredmények ismeretében ezért az operációs rendszerért lelkesednek.
A történet pedig sokkal tovább folytatódik, mint amennyi az expón megmutatkozott belőle. Az Egenera sikeresen értékesíti Linux-alapú kiszolgálóit, amelyek ára kétszázezer és több mint egymillió dollár között mozog.
Az Amazon.com szolgáltatásait nem olyan régen költöztette át a Linuxra, és óriási megtakarítást ért el, amelynek okát az operációs rendszerben látja, és amely kétségkívül hozzájárult a vállalat első nyereséges negyedévéhez.
A 1mage-et (One Image) képviselő *Mary Anne De Young* is annak tulajdonítja cégének utóbbi sikereit, hogy hatalmas mértékben megnőtt az igény a Linux mint unixos termékfelület iránt. A 1mage egyik legnagyobb ügyfele, a Reynolds & Reynolds egyszerre tesz szert haszonra és takarít meg pénzt, miközben a saját (és a 1mage) termékeit autókereskedők ezreinek adja el, linuxos gépeken.
John Gantz az International Data Corporation vállalattól a jövő évre vonatkozó tíz fő előrejelzése keretében lelkesen nyilatkozott a Linuxról.
Arra az esetre, ha ez még mindig nem igazolná elég meggyőzően a Linux hosszú távú kereskedelmi sikerét, álljon itt ez az idézet a Microsoft ügyvezető igazgatójától, *Steve Ballmer*-től: „Úgy gondolom, hogy nagyobb figyelmet kell fordítani azokra a versenytársakra, akik pozíciónkat veszélyeztetik, mint azokra, akikre mi jelentünk veszélyt… Ez alapján a Linux- és a Unix-jelenség vezeti a listát. Én a Linux-jelenséget tartom ránk nézve a legveszélyesebbnek."

Doc Searls

Miért elfogadhatatlan az UCITA ?

Az UCITA, vagyis a Számítógépes Információk Ügyleteit Szabályozó Egységes Törvény (Uniform Computer Information Transactions Act) olyan mintának szánt törvény, amelyet azzal a szándékkal hoztak létre, hogy idővel minden állam elfogadja – ily módon egységesítve a programok felhasználási engedélyének jogi szabályozását. Az UCITA olyan alapértelmezés szerinti szabályokat tartalmaz, amelyek akkor lépnek életbe, ha egy terjesztési engedélyből lényeges feltételek kimaradnak. Az UCITA további célja, hogy meghatározza, mely terjesztési feltételek közérdekellenesek: ezek még akkor sem lépOhetnek érvénybe, ha az engedély tartalmazza őket.
Az UCITA sok tekintetben hasonlít az Egységes Kereskedelmi Törvényre (Uniform Commercial Code – UCC). Az UCC olyan szabályozást állít fel, amellyel megakadályozható, hogy egy kereskedő selejtes árut sózzon rá a gyanútlan vásárlókra. Ebben a szellemben nyilvánították ki az UCITA első változatai is, hogy a közérdekkel ellentétes, ha egy program felhasználói engedélye kizárja a forgalmazhatóságra és a meghatározott célú felhasználhatóságra vonatkozó hallgatólagos jótállást.
Az UCITA az engedély tartalmától függetlenül megköveteli, hogy a terjesztő jótállással nyújtson biztosítékot arra az esetre, ha a program nem bizonyul alkalmasnak olyan egyszerű feladatok ellátására, amelyekre szánták, illetve amelyeket leírása és hirdetési anyagai ígértek. Amennyiben a jótállásban foglaltak nem teljesülnek, a kártérítési igény jelentős lehet. Adott körülmények között a felhasználó a teljes okozott kárt megtéríttetheti, a program várt értéke és a tényleges értéke közötti különbséget, valamint akár a járulékos és következményként fellépő károkat is.
A nyílt forrású programok készítői és terjesztői nem engedhetik meg maguknak az ilyen hallgatólagos jótállás biztosítását. Ha a programot a forráskóddal együtt átadják, hogyan térítheti meg a nyílt forráskód engedélyének kibocsátója a jótállással járó költségeket? Ez az oka annak, hogy minden nyílt forrású program terjesztési szerződése kimondja, hogy a program úgy, ahogy van, mindenféle jótállás nélkül áll rendelkezésre.
Maryland az egyike azon kevés államnak, amelyek elfogadták az UCITA-t. A marylandi törvény hallgatólagos jótállásról szóló része elfogadhatatlan volt a Nyílt Forráskód Közösség számára. Érveink meghallgatása után a helyzet rendezése érdekében a marylandi törvényhozás a következő függelékkel egészítette ki az UCITA-t:
„A jótállás (amely a forgalmazhatóságra és a meghatározott célú felhasználhatóságra vonatkozik) nem érvényes a számítógépprogramra, ha térítésmentes (1.) a forráskód, (2.) a másolatok készítése, illetve azok használata, (3.) a módosítás és (4.) a számítógépprogram terjesztése.
Az UCITA-t az egyes államok egyenként fogadják el. Hogy ne kelljen minden esetben külön ezzel az üggyel foglalkozni, a Nyílt Forráskód Közösségének képviselői egységes törvényszöveg kiegészítését tűzték ki célul. A marylandi törvény szövegét ajánlották elfogadásra az Államok Törvényeinek Egységesítése Megbízottainak Nemzeti Gyűlése (National Conference of Commissioners on Uniform State Laws, NCCUSL) számára, amely az UCITA szerzője. Az NCCUSL ehelyett a következő szövegű indítványt fogadta el:
„(a) Kivéve azokat az eseteket, amikor a (b) alpont lép életbe, a jótállás (amely a forgalmazhatóságra és a meghatározott célú felhasználhatóságra vonatkozik) nem érvényes a számítógépprogramra, ha a kibocsátó a program egy példányát a felhasználó számára ingyenesen hozzáférhetővé teszi, biztosítva a program használatának, másolatok készítésének, a program módosításának vagy másolatok terjesztésének a jogát.
(b) Az (a) alpont nem érvényes, amennyiben egy számítógépprogramot egy árucikk részeként adnak el vagy adnak bérbe, vagy amennyiben a felhasználó nem programfejlesztő."
Az „és" kicserélése „vagy"-ra az (a) alpont vége felé, illetve annak a követelménynek az elhagyása ugyanebben a mondatban, hogy a forráskódnak hozzáférhetőnek kell lennie, egyaránt nagyon fontos változtatás. Ez azt jelenti, hogy az olyan cégek, amelyek ingyenes programjaikat jogdíjas programokkal egy csomagban kínálják – ahogyan azt például a Microsoft teszi az Internet Explorerrel – jogosultak a jótállás alóli mentességre, holott a nyílt forrású programokkal szemben támasztott követelményeket nem teljesítik. Ez az egész függelék voltaképpeni célját kiforgatja.
A (b) alpont hozzáadása szintén veszélyes csapda az elővigyázatlan jogértelmező számára. Az alpont második része azt jelenti, hogy a jótállás hiánya akkor megengedett, ha a programot más programfejlesztőknek adják át, de amint a programot valódi felhasználók vagy ügyfelek használják, a hallgatólagos jótállás érvénybe lép. Köszönjük, de ebből nem kérünk.
2001. november 13-án az Államügyészek Nemzeti Társasága (National Association of Attorneys General) egy harminckét államügyész által aláírt beadvánnyal fordult az UCITA Készenléti Tanácsához. Ez a beadvány általános kifogásokat fogalmaz meg az UCITA-val kapcsolatban, de semmiféle függelék elkészítésére nem tartalmaz lényegi javaslatokat, amely a törvényt a számítógépes információval kapcsolatos ügyletek általános szabályozására alkalmassá tehetné.
Mindaddig, amíg az UCITA nem felel meg az igazságossággal és használhatósággal szemben támasztott követelményeknek, ébernek kell lennünk minden egyes államban, hogy megakadályozzuk e hibás törvény elfogadását.

Linux Journal 2002. június, 98. szám

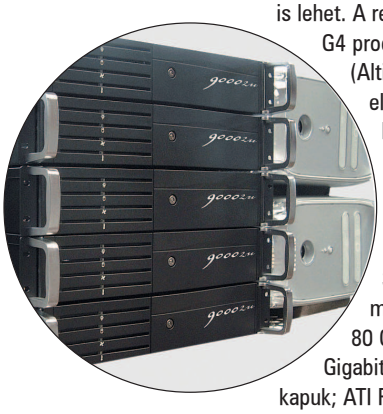
Lawrence Rosen (www.rosenlav.com) magángyakorlatot folytató jogász a kaliforniai Redwood Cityben. A Nyílt Forrás Kezdeményezés (Open Source Initiative) ügyvezető igazgatója és jogtanácsosa (➲ http://www.opensource.org).

Új termékek

3DBOXX R1 és RenderBOXX R sorozat

A digitális tartalom előállítását segítő rendszereket építő BOXX Technologies bemutatta a 3DBOXX R1 sorozatú munkaállomásokat és a RenderBOXX R sorozatú megjelenítő rendszereket. Mindkét rendszerben a QuantiSpeed-felépítéssel és a Smart MP technológiával bíró AMD Athlon MP 2000+ processzor dolgozik. A gépek kétprocesszoros kivitelben, nVidia Quadro4 XGL grafikus kártyával kaphatók, és fő alkalmazási területük 3D-s tartalom és mozgókép létrehozása és megjelenítése népszerű 3D-s programok (Maya, Houdini stb.) segítségével. Az AMD Athlon MP processzort nagyteljesítményű többprocesszoros kiszolgálókba és munkaállomásokba tervezték.
Adatok: BOXX Technologies, Inc., 9390 Research Boulevard, Kaleido II, Suite 300, Austin, Texas 78759, telefon: 1-877-877-2699,
➲ http://www.boxxtech.com

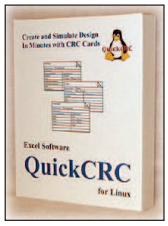
GVS 9000 2U Rack System



A Terra Soft Solutions megjelentette GVS 9000 2U Rack System nevű számítógépét, amely ügyfél, kiszolgáló vagy egy géptelep csomópontja is lehet. A rendszer két 1 GHz-es G4 processzorral van ellátva (AltiVec CPU), valamint előre telepített Yellow Dog Linux 2.0 és Mac OS X operációs rendszert futtatnak. A további összetevők: 1 GB PC-133 SDRAM, két DIMM memória-bővítőhely; 80 GB IDE-merevlemez; Gigabit, FireWire és USB-kapuk; ATI RADEON 7500 kétfejes videokártya; egy 64/66 PCI-bővítőhely; Apple USB-s billentyűzet és egér. A rendszert úgy tervezték, hogy a jelenlegi Macintosh G4-es tornyokkal összehasonlítható legyen.
Adatok: Terra Soft Solutions, Inc., 117 West Second Street, Loveland, Colorado 80537, telefon: 970-278-9243, presales@terrasoftsolutions.com,
➲ http://www.terrasoftsolutions.com

QuickCRC

Az Excel Software bejelentette, hogy elérhető a QuickCRC for Linux, egy olyan objektumközpontú tervezőeszköz, amely gépesíti a CRC-kártyák létrehozását (CRC, azaz osztály, felelősség és együttműködés). A QuickCRC segítségével a programtervezők azonosíthatják az osztályok objektumait, kapcsolatait és egyéb adatait, mielőtt megírnák a kódot. A program a kártyák és a forgatókönyv-objektumok létrehozására diagramokat használ, a projekteket pedig XML fájlformátumba menti. A QuickCRC arra is használható, hogy olyan kártyákat hozzunk létre vagy tulajdonságokat adjunk hozzájuk, amelyek a könnyű programfejlesztési megközelítéshez használhatók, vagy egy nagyobb, UML-t használó modellezési feladat részét képezik. A QuickCRC segítségével a fejlődő terv is jól modellezhető.



Contact Excel Software, 19 Misty Mesa Court, Placitas, New Mexico 87043, telefon: 505-771-3719, e-mail: info@excelsoftware.com,
➲ http://www.excelsoftware.com

Token Ring eszközvezérlők

A Madge Networks bejelentette, hogy elkészültek a Token Ring kártyáihoz szánt nyílt forrású eszközvezérlők. Az új eszközvezérlők programokat tartalmaznak, amelyek támogatják a 2.4.2-es rendszermagot, valamint az előre lefordított meghajtókat – ezek a 2.4.2-es rendszermagon alapuló megvalósításokban használhatók. A következő Token Ring kártyák támogatottak: Madge Smart MK4 PCI-család, RapidFire 3140 PCI-család és a Smart CardBus Mk2 Token Ring kártya. Az eszközvezérlők letölthetők a Madge Networks webhelyéről, a ➲ http://www.madge.com/software címről. A webhelyen a Linux-felhasználók számára fórum is működik.
Adatok: Madge Networks, 1 State Street Plaza, 12th Floor, New York, New York 10004, 800-US-MADGE,
➲ http://www.madge.com

EasiLiX SM

Az EasiLiX SM hálózati biztonságot és rendszerfelügyeletet segítő program. Tervezése során fontos szempont volt a könnyű és gyors telepíthetőség és a karbantarthatóság. Az EasiLiX a következő internetes, illetve belső hálózati (intranet) szolgáltatásokhoz tartalmaz segédeszközöket: DNS, web, e-mail, Squid-proxy, FTP és DHCP. Az EasiLiX egyetlen IP-címet oszt meg a hálózat összes gépe között, és a hálózat a

EASILIX

beépített DNS- és DHCP-kiszolgálók segítségével állítható be. Az EasiLiX további jellemzői: együttműködés a 2.4-es rendszermaggal; ReiserFS; ethernet, xDSL, telefonos és kábelmodem támogatása; 128 bites SSL és OpenSSL támogatása; Samba fájl- és nyomtatómegosztás más operációs rendszerekkel; MySQL-adatbázis; és a PHP, a Perl, valamint a Python parancsnyelvek támogatása.
Adatok: Easilize Software Solutions, e-mail: info@easilize.com,
➲ http://www.easilize.com

Lifix Go! 2.0

A Lifix Go! 2.0 olyan programcsomag, amely a különböző IP-hálózatok között teszi lehetővé az átjárást, például a 802.11a/b/g, ethernet- és cellahálózatok (mint a GPRS és a CDMA 2000). A Lifix Go! részei: mobilkiszolgáló, amely az idegen és a saját ügynökből áll, és a mobilügyfél programja. Az RFC3220-ban rögzített mobil-IP-szabvány teljes mértékben támogatott, akárcsak a több hálózati kártya (az összes felületen) és sokféle VPN-ügyfél. Minden programösszetevő képes RADIUS-hitelesítésre, engedélyezésre és elszámolásra. A mobilügyfél támogatja a Diameter-hitelesítést, engedélyezést és elszámolást is.
Adatok: Lifix Systems Oy, Yliopistonkatu 5, 3rd Floor, FIN-0010 Helsinki, Finland, e-mail: info@lifix.fi,
➲ http://www.lifix.fi

Linux Journal 2002. június, 98.

A hónap szakmai tanácsai

Halott Adathordozó Projekt, Banyan Főosztály

Több HS-8/112-es szalagunk akad még a régi Banyan hálózatos időkből. Egy Exabyte 8200-zal írtunk rájuk. Az adatokat egy linuxos gépen SCSI Exabyte 8505 szalagos meghajtóval próbálom meg visszaolvasni. A `tar` szerint nem `tar`-archívum van rajta. Létezik-e olyan szabadon hozzáférhető módszer, amellyel az adatokat Linux alatt le lehet szedni a szalagról?
Craig Johnson, cjohnson@eypae.com

Nem egyszerű feladat. Létezik néhány dokumentum, amely a más rendszerekkel készült szalagok Linux alatti beolvasásával foglalkozik. Talán egyik sem alkalmazható közvetlenül a te esetedben, de jó tippeket adhatnak és további utalásokat tartalmaznak.
A ➲ http://meteora.ucsd.edu/~pierce/linux_tape.html címen más Unix-rendszerekkel készült szalagok beolvasásáról tudhatsz meg többet. Az alábbin pedig egy másik régi, de értékes HOWTO-fájl található, amely jó ötleteket kínálhat számodra is:
➲ http://www.linux.org/docs/ldp/howto/Ftape-HOWTO.html.
Az első dolog, amit ki kell derítened, a szalagra mentett adatok formátuma. A meghajtó oldaláról közelítve érdemes megtudni, hogy írás közben volt-e valamilyen adattömörítés.
Az operációs rendszer és a programok oldaláról nézve az adatformátum valamilyen elterjedt Unix-formátum lehet, mint a `tar`, a `cpio` vagy a `dd`; másfelől megeshet, hogy a szalagon az adatok különleges szabadalommal védett formátumban rejteznek. Ki kell deríteni azt is, hogy többkötetes mentésről volt-e szó (az adatok egynél több szalagon helyezkednek-e el), vagy sem?
Feltéve, hogy az adatmentésre valamelyik Unix-parancsot használták, megkereshetjük az eltéréseket az írásra használt gép és az olvasásra használt gép parancsainak viselkedése között. Például Linux alatt a `tar` a `-o` kapcsolóval vehető rá arra, hogy a régi (V7) adatformátumban próbáljon írni.
Felipe Barousse Boué, fbarousse@piensa.com

Ez bekerül az osztálynaplóba!

Egy iskolai rendszeren dolgozom, és szükségem lenne egy olyan adatbáziskezelőre, amely lehetővé teszi a következőket:

1. legyen egy rendszergazda, aki a weben keresztül (webböngésző használatával) a távolból dolgozhat,
2. a tanárok (felhasználók) beléphessenek (webböngészőn keresztül), és adatokat rögzíthessenek az adatbázisban.

Maurice Pelletier, mpelletier@sad39.k12.me.us

Javaslom, vizsgáld meg a Zope-ot (➲ http://www.zope.org). A Zope webalapú tartalomkezelő keretrendszer, nagymértékben bővíthető és jól működik. Természetesen meglévő alkalmazásaidat át kell költöztetni vagy át kell írni a Zope alá.
Felipe Barousse Boué, fbarousse@piensa.com

Alagutazás a belső hálóra

Lehetséges-e felépíteni olyan SSH-kapcsolatot, hogy a HTTP-tartalom munkahelyem belső webhelyéről egy alagúton keresztül az otthoni gépem böngészőjére jusson?
Michael Kaneshige, mjkanes@comcast.net

Egyetlen webhely esetén saját géped egyik helyi kapuját átirányíthatod a távoli gépen elhelyezkedő weboldalra, a következő SSH-kapcsolat segítségével:

```
ssh -L1234:intranetserver:80
```

Ezután böngésződbe töltsd be ➲ *http://localhost:1234/* címet. Ne feledd, hogy ezzel megsértheted munkahelyed biztonsági irányelveit. Először őket kérdezd meg a dologról.
Marc Merlin, marc_bts@valinux.com

Ha több belső webhelyet szeretnél kívülről elérni, gépedre telepíts egy Apache kiszolgálót a `mod_proxy` modullal, hozd létre a fenti SSH-alagutat, és webböngésződ proxyját állítsd be a `localhost` 1234-es kapujára. Az Apache `mod_proxy` a ➲ httpd.apache.org/docs/mod/mod_proxy.html címről tölthető le. Ha munkahelyed biztonsági szakembereinek ezzel gondja lenne, mondd meg nekik, hogy mi mondtuk, ez így rendben van.
Don Marti, dmarti@ssc.com

Ethernetkártya gyorsítása

Kiszolgálómba egy 3Com 3C905B-TX hálózati kártya van beépítve. A kártya sajnos csak 10 Mb/s sebességgel hajlandó működni, akár Lantech FE-1601, akár DLink DES-1005D kapcsolóval használom. Hogyan kapcsolhatnám hálókártyámat kétirányú 100 Mb/s üzemmódba?
David Desscan, davidcyberletters@rocketmail.com

Próbáld ki a mii-diag programot a ➲ http://www.scyld.com/diag/index.html webcímről. Ez kiírja, hogy a kapcsolódó fél mit hirdet magáról. Ha itt nem látod a 100BaseTx-FD bejegyzést, kapcsolóid nincsenek megfelelően beállítva, vagy a felek között valamiért nem sikerült a beszélgetés. A következő lehetőség a 100 Mb/s FD-üzemmód kényszerítése, ahogyan ezt a ➲ http://www.scyld.com/network/vortex.html webcímen javasolják. Távolítsd el a modult a memóriából, és a következő paranccsal újra töltsd be:

```
insmod 3c59x debug=1
➥options=4 full_duplex=1
```

Marc Merlin, marc_bts@valinux.com

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

Magyarul jobban hangzik!

Mint a 21. oldalon már megírtam, fontosnak tartom, hogy egységes szóhasználat alakulhasson ki végre az informatika világában is. Azt gondolhatnánk, ez természetes igény, és már réges-régen összefogott mindenki, létezik közös szótár, pontosan és értelmesen tudjuk megfogalmazni mondandónkat magyarul is. Igen, ilyen szótárak vannak, de a céltól oly messze vagyunk!

Vannak társaságok, cégek, ahol már „házon belül” megoldották a szóhasználattal kapcsolatos kérdéseket. Többnyire ilyen a „nagytestvér” is, ahol ma már nagyjából egységesen használják a kifejezéseket. Vannak olyan cégek is, ahol ezt a gondot egyszerűen hidalják át: külső csapatnak adják ki a fordítás nehézkes munkáját. Ezek a fordítók pedig előfordul, hogy a számítástechnikát csak felületesen ismerik, az eredményt – gondolom – nem kell ecsetelnem.

Találkozhatunk olyan fordításokkal, amelyek sokkal jobban „fájnak”. Egyik jó barátom a Közgazdaságtudományi Egyetemre jár, most elsőéves. A minap megmutatta azt a kérdéssort, amely a zárthelyi anyagát dolgozza fel. Nem mondom, hogy bármiféle jogalapom lenne leszólni az említett anyagot összeállítók nyelvhasználatát, de az biztos, hogy az elolvasott tíz kérdés közül nyolcra nem tudtam volna válaszolni – ugyanis nem értettem meg őket. Ilyesféle kérdésekkel találkoztam: „Milyen mértékegységben mérjük a számítógép tárolósebességét?”

Ebből is látható, hogy sokszor olyan magyarításokkal, fordításokkal próbálkozunk, amelyek nem „eresztenek gyökeret” a nyelvben. Az említett példához hasonló mondandójú leveleket mi is rendszeresen kapunk, ezért gondoltam arra, hogy olyan külön rovatot indítsunk, amelyben a nehezebb vagy egyelőre még megoldatlan magyarításokat tárgyaljuk ki. Az említett címszavak mind kerülendő szavak. Természetesen a szavak jelentése nagyban függ a szövegkörnyezettől, így fontos megfigyelnünk, hogy éppen milyen értelemben kerülnek elő. A javasolt fordítást dőlt betűkkel szedtük.

Ha valakinek kérdése, javaslata, ötlete van, szeretettel várom a levelét!

adminisztráció
Használata többértelműsége miatt zavaró (rendszerfelügyelet, irodai munka stb.). Ajánlott helyette a rendszerfelügyelet, vagy röviden ☞*felügyelet*.

applikáció, applet
Ajánlott helyette az ☞*alkalmazás* vagy a ☞*program* (ahol nem zavaró), az applet helyett sajnos nem tudunk jobbat, mint a máshol már használatos ☞*kisalkalmazás*.

archív, archívum
Valamilyen tárolt anyag, illetve tárterület. A .tar, .zip stb. állományok esetén a ☞*tárállomány* kifejezést is használhatjuk, levelezőlistáknál már az ☞*irattár* fordítással is találkoztam.

cluster, kluszter, klaszter
Két értelemben használjuk: az egyik, amikor azonos típusú egyszerűbb eszközöket (például merevlemezeket) kötünk össze és kezelünk egyként, a másik pedig amikor gépekből egy erősebb gépcsoportot állítunk össze. Rendszeresen használt fordítás a ☞*fürt (lemezfürtözés, fürtkezelés stb.)*, a második változatra a megkülönböztetés és érthetőség miatt gyakran a ☞*telep (géptelep)* kifejezést használjuk.

definiál, deklarál
Még mindig rendszeresen kapok olyan vádakat, miszerint a definiálást nem lehet lecserélni a ☞*meghatározással, megadással*. Az indok többnyire az, hogy a „definíció egzakt matematikai fogalom, míg a meghatározás nem annyira pontos”. Egy *Popper*-idézettel válaszolnék: „Nem lehet az ember egy kicsit halálos beteg vagy egy kicsit halott.” A deklarálást pedig arra használjuk, amikor előre ☞*bevezetjük, **megismertetjük*** az adott modult, változót stb. egyfajta bemutatás.

root, root dir, rootdisk
Itt is két fogalmat használunk. Az első a ☞*gyökér*, ☞*gyökérkönyvtár*. Arra utal, hogy (az adott program vagy rendszer számára) ez az elérhető könyvtárak (adatok stb.) legfelső szintje (például egy program „bezárható” a ***/home/ftp*** alá, az adott könyvtárat mint főkönyvtárat látja). A fájlrendszer esetén a / könyvtárnak megkülönböztetett neve is van: ☞*főkönyvtár*. A rendszerindításkor használt rootdisket, mely egy (a rendszerindításhoz szükséges) alap könyvtárszerkezetet tartalmaz, ☞*alaplemeznek* (néhol könyvtárfa-lemeznek) hívjuk.

implementáció, platformfüggetlen, cross-platform, multiplatform
Egy adott leírásnak megfelelő program, az adott leírás egy ☞*megvalósítása*. Vannak olyan programnyelvek, melyekkel úgy írhatunk programot, hogy nem kell a programot futtató rendszerre figyelnünk (ezek a ☞*felületfüggetlen programok*), illetve ugyanaz a program több rendszeren is képes futni (☞*több felületen futó,* ☞*hordozható,* ☞*átvihető*).

quota, kvóta
Egy adott felhasználó által a lemezen elfoglalt terület méretének felső határa (☞*tárkorlát*).

mount, unmount
A fájlrendszerbe egy újabb eszköz beillesztése, ☞*befűzése*. Használják még a csatolás szót is, hátránya, hogy rengeteg értelemben előfordul. A művelet ellenpárja a ☞*leválasztás*.

stackable, stackkelhető
Olyan (hálózati) eszköz, amelynek teljesítménye azáltal bővíthető, hogy további azonos típusú eszközöket kapcsolunk hozzá. Például két huszonnégykapus hálózati kapcsolót ☞*toronyba építhetünk* (☞*összefűzhetünk*), az eredmény pedig egy negyvennyolckapus kapcsolóként működő egység lesz.

szkript, script, shellscript
A script olyan ☞*parancsállomány*, amelynek futtatásához egy értelmezőre van szükség. A futtatókörnyezet külön rendszerként is előfordulhat (mint például a Perl esetében), de az is megeshet, hogy egy adott program vagy héj szabálya szerint az adott héjon belül kerül futtatásra (☞*héjprogram*).

Szy György
a Linuxvilág főszerkesztője. Mindenki véleményét és levelét örömmel várja az alábbi levélcímen: Szy.Gyorgy@linuxvilag.hu

Elemi erők – természeti előnyök

Doc a nyugati területek geológiáját metaforaként használva a Linuxnak a civilizált világ infrastruktúrájában betöltött szerepén elmélkedik.

Ezeket a sorokat a United légitársaság Chicagóból Los Angelesbe tartó Boeing 777-es repülőgépének üzleti osztályán, az egyik székben ülve írom. Balomon egy LCD-képernyő nyúlik ki a kéztámlából. A képernyőn egy térkép látható, amely megmutatja, mit látok, ha a gép jobb oldalán kinézek az ablakon.

A térképet váltakozó nézetekben vehetem szemügyre, akárcsak egy repülőszimulátor esetében. Most éppen azt mutatja, hogy a Sziklás-hegység felé tartunk Fort Collins érintésével, a Colorado állambeli Denvertől északra. Mintegy 30 mérfölddel délebbre található a Wyoming állambeli Cheyenne, amely innen egy repülőtér kifutópályáit körülvevő utcákból és épületekből álló lapos mintának látszik. A város főutak és vasútvonalak csomópontjában fekszik, amelyek – a csaknem állandóan fújó szelek által teljesen simára alakított tájon – halovány vonalaknak tűnnek. Néhány millió évvel ezelőtt az ablakból látható táj nagyrészt olyan lapos volt, mint Nebraska, amikor azonban a hegyek felemelkedtek a síkságokból, a puha talaj elvesztette puha felső rétegét, s ebben a szélnek valószínűleg nagyobb szerepe volt, mint a folyóknak, amelyek munkája sokkal nyilvánvalóbb. Mindez nem csak felületesen érdekel engem, ugyanis az elmúlt néhány évben belemélyedtem ***John McPhee*** műveibe, aki ugyanazt műveli a geológiával, amit Shakespeare a szerelemmel. McPhee számára nincs olyan szikla, melynek a története túlságosan unalmas lenne, és ezt minden egyes lapon, minden egyes könyvében be is bizonyítja.

McPheenek az amerikai geológiát érintő témájú könyveit 1998-ban egy vastag munkában – rövidített formában – összefoglalták, és a „Régi világok krónikája” címet kapta. El is nyerte vele a jól megérdemelt Pulitzer-díjat. Kedvenc könyvem tőle a „Síkságok felemelkedése” (Rising from the Plains), amely két mesét sző egybe: Wyoming távoli múltjának történetét, és ***David Love*** geológusét, aki ennek legutóbbi 89 évében élt. David Love, aki egy gyér termést hozó tanyán nőtt fel az állam kellős közepén, a Földtani Szemle 1955-ben és 1985-ben kiadott Wyomingról készített térképeinek a készítője volt. Kutatásait egymaga végezte, nagyrészt terepen – életének mintegy negyedében szabad ég alatt aludt.

A McPheehez és Love-hoz hasonló emberekben engem az vonz, hogy jó érzékkel kínálnak szilárd alapokon álló nézőpontot egy olyan területre nézvést, amelynek a Linux és a számítógépipar érettebbé válásával egyre alapvetőbbé kell válnia. Ez pedig az infrastruktúra – illetve kedvenc kifejezésemmel élve –, az interstruktúra. A kifejezésen számítógépes és kommunikációs környezetünk legalapvetőbb szintjét értem. A Linux jelen van a még alacsonyabban szinten levő és egyetemesebb jelentőségű környezet fölött, melyet Internetnek nevezünk. Minden egyes kódrészlet, amelyet hozzáteszünk vagy megváltoztatunk, olyan szilárd anyaggá alakul át, amelyből civilizált világunkat építjük fel.

A földtanban a „megbízható” kifejezéssel a sziklára szokás utalni, amely (el)mozdíthatatlan. Építhetünk rajta vagy belőle házat, és bízhatunk benne, hogy nem omlik össze, ha a meredek részén másznunk fel. Nincs benne semmi titokzatos, rejtegetnivaló sem. Ugyanez igaz a programkódra is. Az infrastrukturális programkód természeténél fogva megbízható. Ezen kívül szabadon felülvizsgálható és továbbfejleszthető. Az infrastrukturális programkód fejlesztésekor használt értelmi és alkotó folyamatok sem kevésbé természetesek, mint azok a földtani erők, amelyek a mészkövet márvánnyá alakítják.

A megbízhatóságnak egy másik szempontja is létezik. David Love szerint „az emberi környezet, a jó és a rossz is a sziklával kezdődik, összekapcsolva a két másik létszükséglettel: a vízzel és a levegővel. Ha tönkretesszük valamelyiket e három alapvető létszükséglet közül, az emberiség nagy bajba kerül.” Most cseréljük ki a „szikla” szót „Internetre”, és az „emberiség” szót „számítástechnikára”, és rögtön jobban megértjük, Love mire gondol.

Az infrastruktúrától függünk. És mivel a természetben számos helyütt bőségesen megtalálható, sokan értik közülünk a működését és a jelentőségét.

A Linux „átlátszósága” a Linux természetétől fogva adott infrastrukturális előnye a Windows-rendszerrel szemben. Még ha a Windows százszázalékosan megbízhatóvá válik is, továbbra is átlátszatlan marad mindaddig, amíg forráskódja zárt lesz. Építs belőle bármit, amit akarsz, csak alapkőként ne használd.

Ez nem azt jelenti, hogy a kereskedelmi fejlesztőknek semmi keresnivalójuk sincs az infrastrukturális alapkövek megalkotásában. A SOAP és XML-RPC (mindkettő nyílt szabvány) kereskedelmi fejlesztője, ***Dave Winer*** úgy véli, hogy „ne azt kérdezd, mit tud tenni érted az Internet, hanem azt, mit tudsz te tenni az Internetért”. A Linux folyamatos továbbfejlesztése lehet az egyik válasz erre a kérdésre.

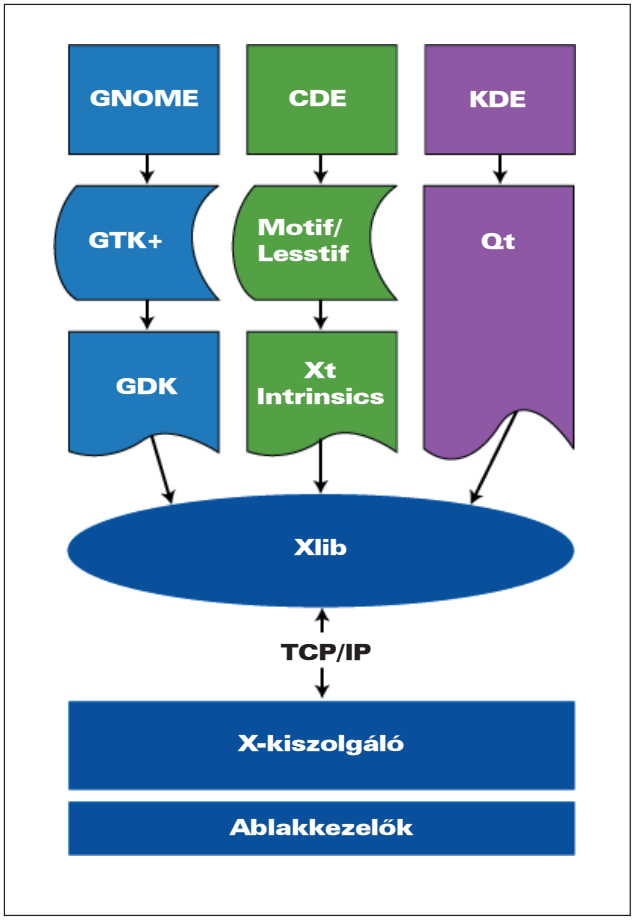
Amint lassan megkezdjük a leszállást Los Angelesben, azon kapom magam, hogy arra gondolok, hosszú távon (a számítógépes infrastruktúra megszilárdulásához szükséges időszakban) a valódi harc nem a Linux és a Windows között zajlik majd, hanem az infrastruktúra természetét tiszteletben tartók és azt figyelmen kívül hagyók között. Akik tiszteletben tartják, azok az Internetet tekintik az egyedüli olyan felületnek, amely méltó erre a szóra. Akik nem tartják tiszteletben, azok a Hálót csak egy újabb csővezetékrendszernek tartják. Velük vívandó harcunk azoknak az üzleti modelleknek a védelmét szolgáló szabályozásról fog szólni, amely modelleknek szükségük van a csővezeték feletti ellenőrzésre, és amelyeket az Internet hatékonysága azzal fenyeget, hogy szükségtelenné válnak. De győzni fogunk, mert a természet a mi oldalunkon áll.

Doc Searls
(doc@ssc.com)
a Linux Journal szerkesztője és a Cluetrain Manifesto társszerzője.

Könnyed programfejlesztés KDE alá!

Aki e cikket elolvassa, és rendelkezik némi programozói tapasztalattal, minden szükséges információhoz hozzájuthat, ami egy linuxos asztali alkalmazás elkészítéséhez kívánatos.

Számos linuxos alkalmazásfejlesztő eszközkészletből (toolkits) választhatunk. Néhányan úgy gondolják, ez már a Linux bukásának kezdetét jelenti, mások szerint éppen ez a legnagyszerűbb képessége. Én az arany középutat választom, és úgy vélem, jó megoldás, de csak akkor, ha olyat választunk, amely testreszabottan felel meg igényeinknek.
Linux alatt a legtöbb grafikus munkafelület (GUI) az X-en alapul, azaz egy olyan kiszolgálóalapú szerkezeten, amely a hálózati gépek számára grafikus alkalmazásaik megosztását teszi lehetővé. Az X szempontjából az ügyfél az az alkalmazás, amely grafikus kimenetét az X-kiszolgálónak továbbítja. Az X-kiszolgáló az alkalmazások kimenetét saját helyi eszközökről (avagy virtuális eszközökről, de írásomban erre terjedelmi korlátok miatt nem térek ki) fogadhatja. A legtöbb esetben az X-kiszolgáló és az X-ügyfél ugyanazon a gépen fut, ennek ellenére ilyenkor is a kiszolgálóalapú modellt használja.
Az X-ügyfelek készítéséhez felhasználható alapszintű eszközkészlet neve Xlib. Az Xlib azonban túlságosan alapszintű és nehézkesen használható ahhoz, hogy önmagában alkalmazások felépítésére használhatnánk fel. Ennek eredményképpen rengeteg Xlibre épülő eszközkészlet készült, amelyekkel X alatt könnyebben lehet GUI-alkalmazásokat létrehozni. Ráadásul ha alkalmazásunkat ilyen magas szintű eszközkészletben készítjük el, arra sem kell figyelnünk, hogy valójában hálózati alkalmazást készítünk (ami grafikus kimenetét egy kiszolgálóhoz küldi).
A két legnépszerűbb Xlibre épülő nyílt forrású eszközkészlet a Qt és a GTK+ – a KDE és a Gnome is ezekre épül. A Motif szintén népszerű eszközkészlet (ez ugyan nem nyílt forrású, de Lesstif néven ingyenes változata is létezik). Az *ábrán* bemutatott diagramon ezeket (és néhány más) rendszert, valamint egymás közötti kapcsolataikat láthatjuk. Minél lejjebb helyezkedik el valami a diagramon, annál alacsonyabb szintű API-ról van szó. A magam részéről több okból is a KDE/Qt megvalósítást kedvelem, legfőképpen azért, mivel a jó felhasználói felületre összpontosít, ugyanakkor tiszta és jól megtervezett API.



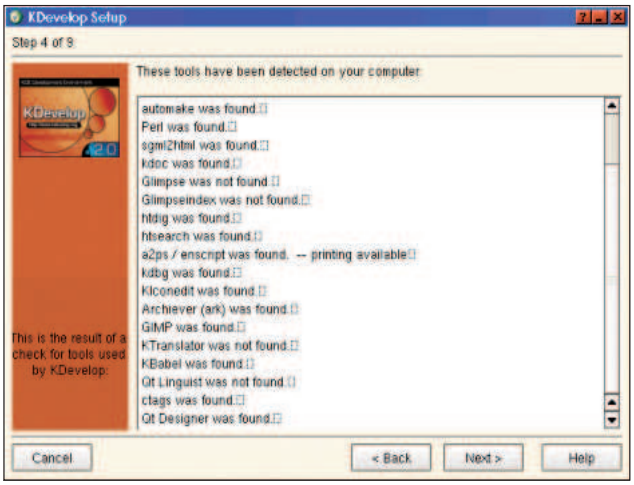
Rokonsági kötelékek a KDE, a Gnome és egyéb eszközök között

Előfeltételek

Ebben a cikkben egy alkalmazást (számológép) fogunk az alapoktól felépíteni, hogy megmutassam, milyen gyorsan és egyszerűen létre lehet hozni. A lépések követéséhez szükségünk lesz néhány eszközre. A legfontosabb a KDevelop, a 2.0.2-es változatot használtam KDE 2.2.2 alatt. Ha a KDevelop telepítő varázslóját használjuk, a rendszer figyelmeztet az esetlegesen hiányzó függőségekre. Látogassuk meg a KDevelop honlapját (➲ http://www.kdevelop.com), vagy nézzük át a terjesztésünkhöz adott programok listáját, és ha a KDE szerepel a terjesztésünkben, telepítsük a KDevelopot.

KDevelop

A legtöbb alkalmazásfejlesztési útmutató megpróbál fejlesztőkörnyezet-független maradni. Én ezt a hagyományt két okból is figyelmen kívül hagyom: a KDevelop ingyenes és a KDE-vel együtt érkezik; a KDdevelop egy már eleve könnyen kezelhető API-t még egyszerűbbé tehet, ráadásul a KDevelop varázslói segítik a fejlesztőket, hogy betarthassák a KDE felhasználói felületekre vonatkozó szabványait. Azoknak, akik nem nagyon szeretnek `Makefile`-okkal bíbelődni vagy beállításfájlokat készíteni (esetleg az egyszerűbb `automake` és `autoconf` sincs ínyükre), nagy segítség lehet a KDevelop, hiszen mindezt megteszi helyettük. Azon szakik számára, akik mégis szeretnek `Makefile`-okkal bíbelődni, a KDevelop ezt is lehetővé teszi. KDevelop-alkalmazásunk a hagyományos `./configure; make; make install` megoldás szerint is elkészíthető. Más szóval a KDevelop használatát tetszés szerint bármikor lehetőségünk nyílik megszakítani.
Amikor első ízben indítjuk el a KDevelopot, az végigvezet bennünket a telepítővarázslón. A varázsló legfontosabb feladata a



1. kép A KDevelop telepítésvarázslója

függőségek vizsgálata (*1. kép*). Vegyük a fáradtságot, és nézzük végig a kimenetet – ha hibajelentéseket találunk benne, segítsünk a gondon, és próbáljuk meg a hiányzó könyvtárakat feltelepíteni. Ezt legkönnyebben a ➲ http://www.rpmfind.com segítségével tehetjük meg (amennyiben RPM-alapú rendszert használunk). Minden hiányzó könyvtárhoz végezzünk keresést az rpmfind honlapon. A találatokból válasszuk ki a terjesztésünknek megfelelő RPM-csomagot, majd telepítsük. Ha ezzel végeztünk, futtassuk újra a KDevelop varázslóját (a `kdevelop --setup` parancsot gépeljük be, vagy a K menüből a ***KDevelop Setup*** nevű menüpontot keressük ki). Ha még mindig lennének hiányzó elemek, a fenti lépéseket szükség szerint ismételjük meg.

Kalculate

Az általam felépített számológép igen egyszerű, és csak a számtani alapműveletekre képes. Az egész elkészítése mindössze két órát vett igénybe (na jó, három órát, de nem siettem). Úgy döntöttem, hogy Kalculatornak nevezem el, de mivel már létezik ilyen nevű alkalmazás, végül a Kalculate név mellett döntöttem. Nem használtam, sőt meg sem néztem a KDE KCalc nevű beépített számológépének forráskódját, amely természetesen több matematikai szolgáltatást nyújt, mint az enyém. Mindig is úgy éreztem, jó lenne, ha a KDE tartalmazna egy egyszerű számológépet, ezért szívesebben látnék néhány felhasználóbarát képességet a Kalculate-ben a legújabb matematikai csodák helyett. Miután elolvastad a cikket, meghívlak egy élő nyílt forrású fejlesztés boszorkánykonyhájába – jöjj, és csatlakozz a csapathoz, és adj újabb képességeket a programhoz! Különleges előnyöket élvezhetsz, ha megemlíted, hogy e cikk közvetítésével jutottál hozzánk (➲ http://sourceforge.net/projects/kalculate).

A Document-View Model

Nem fogunk új projektet kezdeni, hanem kiaknázzuk az Internet nyújtotta lehetőségeket, és elkezdjük használni a Kalculate-et. A következő parancsok kiadásával töltsük le a Kalculate kódját a hivatalos CVS-kiszolgálóról (feltételezve, hogy írási joggal rendelkezünk, készítsünk egy ***/usr/local/src*** nevű könyvtárat, amennyiben esetleg még nem rendelkeznénk vele):

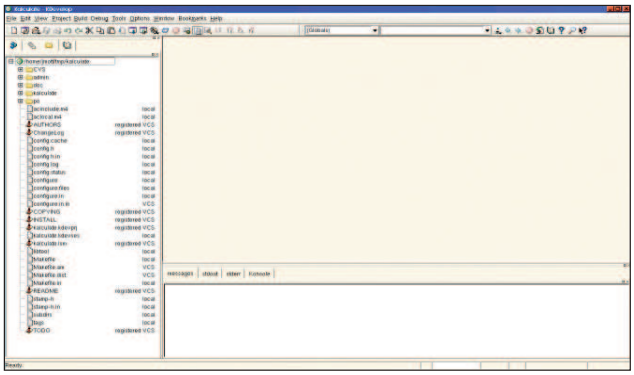
```
cd /usr/local/src
cvs -d:pserver:
➥anonymouscvs.kalculate.sourceforge.net:
➥/cvsroot/kalculate login
```

Ha a parancs jelszót kér, egyszerűen üssünk ENTER-t, ugyanis nincs szükség jelszóra.

```
cvs -z3 -d:pserver:anonymous@cvs.kalculate.
➥sourceforge.net:/cvsroot/
➥kalculate co kalculate
```

Amennyiben az alkalmazást mi magunk szeretnénk felépíteni, nincs szükségünk a forráskódra. Mivel azonban most nem szándékozom soronként leírni a folyamatot, lehetséges, hogy mégis célszerűbb a forráskódot letölteni, és frissen szerzett tudásunkat új képességek hozzáadásával vagy egy új alkalmazáson kipróbálni.
Az általunk alkalmazott tervezési minta az úgynevezett Document-View modell. Három alkalmazásspecifikus osztállyal nyitunk, ezek a következők: a `KalculateApp`, a `KalculateDoc` és a `KalculateView`. E három osztály jön létre, ha a projektet KDeveloppal frissen hozzuk létre.
A Kalculate esetében a sablon által felajánlottakhoz további osztályokat már nem adtam – azért tettem így, mert ily módon könnyebb bemutatni a tervezési mintát, illetve azt, mennyi mindent megold számunkra a KDevelop. A működő számológép előállításához mindössze három osztályt kellett módosítanom, néhány ikont hozzáadnom, és voil□!
A Document-View legnagyobb hibája, hogy az üzleti logika a dokumentum végére kerül, és a felhasználói felület részt az utolsó nézet tartalmazza. Elméletben ugyanannak a dokumentumnak több nézete lehet. Általános értelemben véve a „dokumentum" valamilyen „dolog" adott nézete. Természetesen ezt a legkönnyebben valamilyen szövegszerkesztő-szerű dologgal tudjuk összeegyeztetni. Ezen elképzelés szerint a dokumentumobjektum jeleníti meg a szövegfájl minden jellemzőjét és tulajdonságát, amelyben a nézetobjektum (view object) felelős a szövegképernyő megjelenítéséért és a felhasználói felületéért. A vázban azt is észrevehetjük, hogy a mentés, nyomtatás, megnyitás és lezárás eljárásai előre elkészültek a számunkra. E modellnek egyértelműen a szövegszerkesztő-elképzelés képezi alapját. Csakhogy például a Kalculate esetében ez a megközelítés nem mindig működik. A számológép nem igazán használ fájlokat: egy megadott képességkészlettel rendelkezik, és ezt teszi elérhetővé. Ezért – bár üzleti logikánkat a `KalculateDoc` osztály valósítja meg – nem lesz szükségünk a megnyitás, lezárás, mentés és nyomtatás képességekre. Egyelőre a Kalculate forrásában a szükségtelen gyári kódot (vagyis azt, amit a KDevelop írt be) megjegyzésbe tettem, ahelyett, hogy egyszerűen töröltem volna. Ezáltal, ha egyszer mégis szükség lenne rájuk, a vázlatok megmaradnak.
Az üzleti logika felől nézve a számológépnek lehetővé kell tennie a számok begépelését, e számokhoz műveletek hozzárendelését, és az eredményszámot vissza kell szerezni (illetve a műveleteket belül valóban meg kell valósítani). A felhasználói felület szemszögéből gombokra van szükségünk, amelyekkel számokat és műveleteket vihetünk be, illetve megjelenítő felületre, amelyen az eredményt írjuk ki. Feltételezve, hogy a `cvs checkout`-ot a ***/usr/local/src*** könyvtárban végeztük, a ***/usr/local/src/kalculate*** könyvtár fog létrejönni. A következőket gépeljük be:

```
cd /usr/local/src/kalculate
kdevelop  kalculate.kdevprj &
```



2. kép KDevelop projektfájl

A csatolók

```
public slots:
  /**megh vja a repaint()-et minden nØzetre,
   * ami a document objektumhoz
   * kapcsol dik, illetve az a nØzet h vja
   * meg, amelyikben a dokumentum
   * megvÆltozott. Mivel ez a nØzet
   * normÆlisan œjrarajzolja magÆt,
   * kivessz k a paintEvent al l.
   */
  void slotUpdateAllViews(KalculateView
  ➥*sender);

  /** Minden szÆmol gØp szolgÆltatÆs,
   * egy-egy tagf ggvØny minden egyes
   * gombhoz, ahogy az egy val di gØpen
   * is lenne. ValamifØle setNum(int)
   * f ggvØnymegoldÆs helyett ezt a
   * tervezØsi m dot alkalmaztam, nemcsak
   * azØrt, hogy az objektumot
   * biztonsÆgossÆ, hanem hogy szabad
   * t pusœvÆ is tegyem.
   *  AzØrt ezt a megoldÆst rØszes tettem
   *  elınyben, mert  gy a hÆtteret
   *  bÆrmikor anØlk l vÆltoztathatjuk meg,
   *  hogy bÆrmit elrontanÆnk.
   */
  void pressZero();
  void pressOne();
  void pressTwo();
  void pressThree();
  void pressFour();
  void pressFive();
  void pressSix();
  void pressSeven();
  void pressEight();
  void pressNine();
  void pressPlus();
  void pressMinus();
  void pressTimes();
  void pressDivide();
  void pressToggleSign();
  void pressEquals();
  void pressDecimal();
  void pressClear();
```

A fenti parancs megnyitja a projektet. A KDevelop a projektadatokat a ***.kdevprj*** végződéssel kiegészített projektnév nevű fájlban tárolja. Beállításainktól függően valami olyasmit kell látnunk, amit *2. kép* mutat. Ha a bal oldalon nem látjuk a projektfát, ellenőrizzük, hogy a ***View, Tree Tools, Views Files*** be van-e kapcsolva. A ***kalculate*** könyvtárra kattintva a projekt fájlszerkezetét láthatjuk (igen, az ott egy ***kalculate*** könyvtár a ***kalculate*** könyvtár alatt, nem kell megtisztítani a monitort!). Itt található az összes forrásfájl. Számunkra most a ***kalculatedoc.cpp*** és ***kalculatedoc.h*** fájlok a legérdekesebbek. Kattintsunk rájuk duplán. Ez a két fájl számológépünk magja: a `KalculateDoc` osztály kezeli az összes számítást. Ez egy olyan virtuális gép, amelyhez valakinek hozzá kell fűznie az irányítófelületet.
Ha a ***kalculatedoc.h***-t megvizsgáljuk, az osztálymegadásokban hamar felfedezhetünk valami furcsaságot, ugyanis akad köztük néhány, amelyik mintha hibás írásmódot követne. Ez vezet el bennünket a csatolók és jelek (signal) világába.

Csatolók és jelek

Az eseményvezérelt alkalmazásokban, mint amilyenek az asztali alkalmazások, egy olyan alrendszert kell felépíteni, amelynek adott részei bizonyos események bekövetkezésekor meghívhatók. Ez az esemény lehet valamilyen felhasználói cselekmény (az egérmutató mozgatása, egy egérgomb lenyomása, egy billentyű leütése stb.). Eseményt egy másik programelem is okozhat, amely valamilyen állapotot jelez (például az időt, a betelt fájlrendszert). Az alkalmazások egyik általános grafikus eleme (widget) a gomb, mellyel könnyen választhatunk ki eseményeket. A KDE világában az eseményeket jeleknek (signal), az eseményeket kezelő függvényeket pedig ***csatolóknak*** vagy eseménykezelőknek (slot) nevezzük. Fejlesztőként lehetőségünk nyílik jeleket és csatolókat készíteni, illetve megadhatjuk, hogy melyik jel melyik csatolóhoz csatlakozzon. Csatolóinkat beépített jelekhez is csatolhatjuk (és megfordítva), vagy beépített csatolókhoz beépített jeleket rendelhetünk.
A beépítettség jelen esetben azt jelenti, hogy a KDE-t vagy a Qt könyvtárosztályokat használjuk, amelyek előre meghatározott csatolókkal és jelekkel rendelkeznek.
A Qt (és így a KDE) a hibás írásmódot is elfogadja, mivel a szabványos C++ előfeldolgozó `#define` fordítói utasításaival a `signals:` megadásokat `protected:`-del helyettesíti, a `slots` és `emit` megadásokat pedig törli. A kiegészítő utasításformára az előfeldolgozó előtt futtatott `moc`-nak van szüksége.
A Meta Object Compiler (moc) egy olyan program, amely a Qt könyvtárral együtt érkezik, és a `QObject` fejállományain kell lefuttatni, hogy az osztályainkhoz tartozó különleges tagfüggvényeket elkészítse, és így azok `QObject` alosztályokként felhasználhatók legyenek. Minden objektumnak, amely csatolókat és jeleket használ, kötelezően a `Qobject` alosztályának kell lennie, osztálymeghatározásában pedig szerepelnie kell a `Q_OBJECT` makrónak (amint azt a ***kalculatedoc.h*** fájlban láthatjuk is). A `QObject` szuperosztály egyik tagfüggvénye a `connect`. Ezzel a tagfüggvénnyel kapcsolhatjuk össze jeleinket és a csatolókat.
Jelek létrehozásához osztálymeghatározásunk `signals:` bejegyzése alatt meg kell adnunk őket. A jelek esetében valójában nincs szükség további tagfüggvények elkészítésére. Mindössze annyit kell még tennünk, hogy az `emit`-et a jel nevével meghívjuk, és máris az összes hozzácsatolt csatoló meghívódik – a `moc` az összes jeltagfüggvényt elkészíti nekünk.
A csatolókat ezzel szemben létre kell hozni. Csatolójainkat megadhatjuk a `public slot` vagy a `protected slot` alatt, majd a ***.cpp*** fájlban hagyományos tagfüggvényként létrehoz-

hatjuk őket. Ezek után már tetszés szerint csatlakozhatunk hozzájuk, valahogy így:

```
connect(this, SIGNAL( mySignal() ),
➥this, SLOT( mySlot() ) );
```

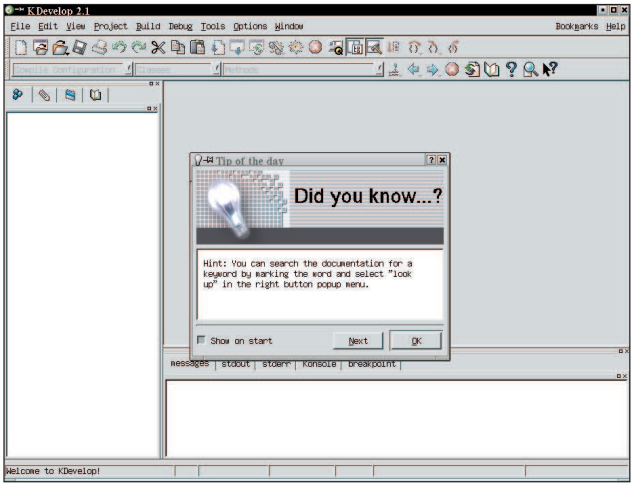
Feltételezve, hogy ugyanazon az osztályon belül van egy `mySignal()` nevű jelünk és egy `mySlot()` nevű csatolónk, a fenti `connect` hívás összefűzi a kettőt, így valahányszor kiadunk egy `emit mySignal()` utasítást, a `mySlot()` meghívódik. Ha csatolónkat valamilyen más objektum jeleihez szeretnénk kapcsolni, az első érték az adott objektum példánya lett volna, a második érték pedig a `SIGNAL()` makró zárójelek között a jel nevével. Ilyen egyszerű. Minden nehéz részt (ideértve fejállományainkon a `moc` futtatását) a KDevelop végzi el helyettünk. Tehát – számológépünkhöz visszatérve – a `KalculateDoc` osztály a csatolók halmaza. Minden csatoló egy-egy olyan műveletnek felel meg, amit a számológéppel elvégezhetünk. A fejállomány az összes csatolót a *listában* látható módon határozza meg. Az egyetlen dolog, ami ezeket a tagfüggvényeket különlegessé teszi, az az, hogy a `public slots:` címke alatt lettek megadva, így a `moc` némi metaadatot készít hozzájuk. A csatolókat ugyanúgy kell megadnunk, mint a hagyományos tagfüggvényeket (vizsgáljuk meg a ***kalculatedoc.cpp***-t, és figyeljük meg e csatolók meghatározásait). A jelek esetében ezt nem kell megtennünk, a `moc` mindenről gondoskodik.

3. kép Kalculate

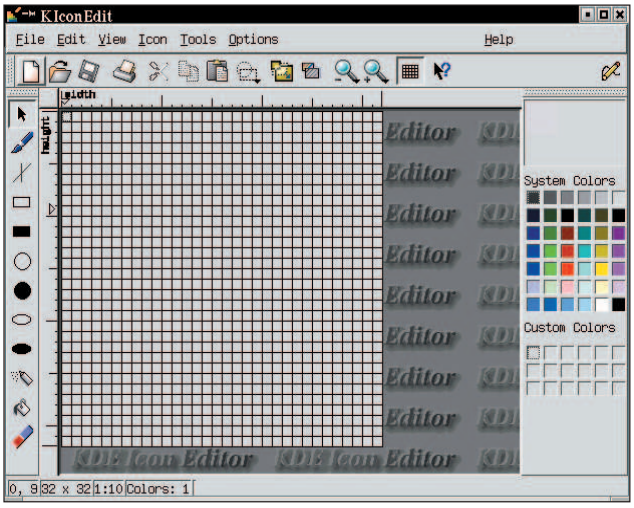
Most már csak egy GUI-ra van szükségünk, ami ezt az osztályt használni tudja. Lépjünk be a `KalculateView`-ba (reményeim szerint a forráskód megtekintése végett már amúgy is beléptél ide). A ***kalculateview.h*** fájl határozza meg a GUI-t létrehozó osztályt. A `protected` kulcsszó alatt láthatjuk, hogy néhány kinézetkezelő, LCD-megjelenítő és pár gomb már meg van adva. A KDevelop beépített sablonkódjához hozzáadtam egy fájlt, a neve ***kalculatesizes.h***, a tartalma pedig itt olvasható:

```
#ifndef KALCULATE_SIZES_H
#define KALCULATE_SIZES_H

#define BUTTON_WIDTH 35
#define BUTTON_HEIGHT 35

#define LAYOUT_SPACING 4

#define MAX_WIDTH (BUTTON_WIDTH * 5)
        + ((LAYOUT_SPACING *2) * 4)
#define MAX_HEIGHT (BUTTON_HEIGHT * 5)
        + ((LAYOUT_SPACING *2) * 4)

#endif // SIZES_H
```

Ez tulajdonképpen számológépünk méreteit állítja be. Nem szerettem volna újraméretezhető számológépet készíteni, ugyanakkor azt akartam, hogy a gombok méretét könnyedén lehessen változtatni. Ezért a méreteket ebben a fájlban adom meg. Ezek után mindössze ezt a fájlt kell módosítani, majd újrafordítani a programot, és máris eltérő gombméretű számológépet kaptunk



4. kép A KDevelop induló ablaka



5. kép Az ikonkészítő felülete

(a jövendőbeli változatok esetleg futásidőben is lehetővé tehetik, ezt azonban bizonyos okok miatt haboztam megtenni). A `KalculateView`-ban található létrehozó függvény (a ***kalculateview.cpp***) kelti életre az alkalmazást, a `setMaximumSize()` meghívása pedig átméretezhetetlenné teszi. A méretszabály szintén segít, ez azonban csupán ajánlás. Itt megállnék egy pillanatra, és ejtenék pár szót a kinézetkezelőkről (layout managers). Minden egyes GUI-elemnek vannak tagfüggvényei, amelyek a méreteit állítják (magasság, szélesség, relatív helyzet stb.). Amikor az alkalmazás elindul, illetve valahányszor átméretezik, nem lenne vicces, ha mindig nekünk kellene megírnunk azt a kódot, amely átméretezi és áthelyezi. A kinézetkezelő nélkül kezdeti állapotban az alkalmazás minden egyes gombja előre rögzített x:y koordinátákra kerülne. Órákat venne igénybe, ha a számológép méretét meg akarnánk változtatni. Ezért az elhelyezési tulajdonságok rögzítése helyett az elemkészleteket kinézetkezelőkben jegyezzük be, amelyek bizonyos szabályokat követnek, például a `QVBoxLayout` az elemeket függőleges oszlopokba rendezi. Valahányszor új elemkészletet adunk a hozzá az `addWidget()` tagfüggvénnyel

```
outerL0ayout->addWidget(output,1);
```

az az új elemkészletet önműködően az előző alá helyezi. A `QHBoxLayout` ezzel szemben az új elemkészletet mindig az előző jobb oldalára helyezné. A második érték az úgynevezett nyújtási arány (stretch factor). A nyújtási arány alapvetően azt határozza meg, hogy ez az elemkészlet mennyi helyet foglaljon el az ugyanebben a kinézetkezelőben található egyéb elemkészletekhez képest. Így tehát ha valamennyi `1`, mindannyian ugyanakkorák lesznek (feltéve, hogy a `setMaximumSize` ezt az értéket nem írja felül). A nyújtási arányok összege adja meg a mértéket. Így ha van két elemkészletünk, és az egyik `1`-es, a másik `2`-es nyújtásaránnyal bír, a második kétszer akkora lesz, mint az első.

Igazán nagyszerű benne, miként a `KalculateView` is példázza, hogy kinézetkezelőkbe másik kinézetkezelőt is helyezhetünk. Ez lehetővé teszi, hogy igen összetett kinézeteket alkossunk. Kalandra fel, próbálkozz egy kicsit a Kalculate kódjával – fedezd fel, mit tudsz alkotni!

Az alkalmazás futtatásához a következő lépéseket kell megtennünk: válasszuk az ***Autoconf*** és ***Automake*** pontot a ***Build*** menüből. Majd válasszuk a ***Build*** menü ***Configure*** pontját. Amikor értékeket kér, a `--prefix=`*`kde alapk nyvtÆrad`*-at adjuk meg. Ez lehetővé teszi, hogy az alkalmazást telepítsük is, ami elengedhetetlen, ha azt szeretnénk, hogy az alkalmazás ikonja megjelenjen. Mandrake-et futtató gépem KDE alapértelmezett könyvtára a ***/usr***. Más terjesztésnél eltérő lehet. Ezután válasszuk a ***Build*** menü ***Execute*** pontját, és az alkalmazás lefordul, majd lefut. A *3. képen* láthatóhoz hasonló látványnak kell fogadnia minket, bár a bal felső sarokban szerplő ikon – amíg fel nem telepítjük – valószínűleg hiányozni fog. A telepítéshez rendszergazdaként a következőket gépeljük be:

```
cd /usr/local/src/kalculate
make install
```

Remélem, mindenkit a helyes irányban indítottam el. Valójában éppen csak a felszínét kapargattuk meg a KDE-vel végezhető feladatok hatalmas tárházának, reményeim szerint azonban sikerült megmutatnom, hogy milyen gyorsan el lehet indulni a KDeveloppal végzett munka rögös útján. Csaknem az összes háttérmunkálatot átvállalja tőlünk, és olyan alkalmazást készít, amely parancssorból lefordul anélkül, hogy a KDevelop szükséges lenne hozzá (ami nagyon jó, ha a kódot terjeszteni szeretnénk). Szívesen várom leveleiteket, amennyiben esetleg kérdésetek lenne, és ezúton is bátorítanék mindenkit, hogy csatlakozzon a Kalculate-csapathoz, és adjon hozzá néhány további képességet.

Linux Journal 2002. június, 98. szám

Jason Mott
(jmott@users.sourceforge.net)
független programozó és tanácsadó. Jelenleg a rochesteri ElementK-nak dolgozik (➲ http://www.elementk.com) New Yorkban, hálózati oktató honlapjuk kialakításában segíti őket. Részidőben Linux-tanácsadó, és ha szabadideje engedi, linuxos asztali alkalmazásokat készít.

Az fsck rendszerellenőrző és -helyreállító eszköz

Tudjuk, hogy rendszerünk sohasem omlik össze, a valóságban azonban a legrosszabb időnként mégis bekövetkezhet. Ha nincs szünetmentes áramforrásunk, elég lehet, ha valaki véletlenül kihúzza a hálózati csatlakozót vagy áramszünet áll be. Ha ez megesik, előfordulhat, hogy a fájlrendszer megsérül. Ha a rendszer a leállás pillanatában éppen adatokat írt egy állományba, jó eséllyel csonka fájl és adatszemét marad vissza. Bárki, aki hosszabb ideje használ Linuxot (vagy Unixot), tanúsíthatja, hogy ezen a területen a rendszernek nincs szégyellnivalója. Az ext2 fájlrendszer az ilyen nehézségek kezelésében alapvetően jobb, a baleset időről időre mégis bekövetkezik.

Amikor valami ilyesmi történik, a rendszer a következő indítási próbálkozásnál észreveszi, hogy a lemezterület nem szabályosan lett kifűzve (gondoljunk csak a Windows ScanDisk programjára).

A program, amelyik a rendellenességet észleli, ugyanaz, mint amelyik a helyreállítást végzi. A program neve fsck (ez egyébként a file system check, vagyis a fájlrendszer-ellenőrzés rövidítése). Az imént az ext2 fájlrendszer melletti érvet említettem azért, hogy most valami érdekeset mutathassak. Az `fsck` parancs nem az fsck programot futtatja. Ez csak a felülete a különböző fájlrendszerek javítóeszközeinek.

Az ext2 fájlrendszerhez tartozó program neve fsck.ext2, bár lemezünkön a program ext2fs néven található. Mivel mindkettő ugyanaz, mindegy, melyiket használjuk.

Mivel az fsck önműködően elindul, nem feltétlenül kell tudnunk róla, hogy kézi indítására is lehetőség van. Ehhez azonban az szükséges, hogy az ellenőrizni kívánt fájlrendszert kifűzzük. A másik lehetőség, hogy a rendszert egyfelhasználós módban indítjuk (`linux single` beírása a LILO parancssorába). Egy fájlrendszer ellenőrzéséhez a következő parancsot használhatjuk:

```
fsck /dev/hda5
```

Ez a legegyszerűbb módja a lemezellenőrzésnek. Amennyiben a lemez kifűzésével nem volt gond, az fsck visszatér. Ha mindenképpen végre kívánjuk hajtatni az ellenőrzést, a `-f` kapcsolót kell használnunk.

```
fsck -f /dev/hda5
```

Ha a lemezzel gondok merülnek fel, és az fsck nem biztos benne, hogy mit tegyen, a változtatások előtt a felhasználó megerősítését kéri. Ha mindent rá akarunk hagyni, a `-y` kapcsolót használhatjuk, ezzel minden esetleges kérdésére igenlő választ adunk:

```
fsck -f -y /dev/hda5
```

Amennyiben az fsck úgy találja, hogy egy fájlrészletet képtelen visszafűzni a rendszerbe, a fájldarabot a *lost+found* könyvtárba helyezi. Mivel az fsck szakértő módon végzi feladatát, általában semmit sem fogunk találni benne. Ennek ellenére, ha rendszerünk nem szabályosan lett leállítva, és arra kényszerült, hogy az fsck-t futtassa, talán érdemes megnézni a *lost+found* könyvtárat, hátha mégis valami „elveszett és megkerült".

Részlet Marcel Gagné: Linux-rendszerfelügyelet című könyvéből

Közösen használt fájlrendszerek titkosítása

Mick a BestCrypt nevű nyílt forrású alkalmazást fogja megvizsgálni, amely titkosított kötetek Windows- és Linux-felületek közti megosztását teszi lehetővé.

A személyi titkosítás szószólói számára az idei március fekete hónap volt. A Network Associates hivatalosan bejelentette, hogy felhagyott a PGP Desktop (minden számítógépre külön telepített titkosítási eszköz) támogatásával – egyszerűen szólva ez a manapság használatban lévő legnépszerűbb, legérettebb, és leghasználhatóbb végfelhasználói titkosítási eszköz. Kimondhatatlanul nehezen viselem, ha egy kereskedelmi termék nagyszerűségét kell elismernem egy ingyenes termékkel szemben, jelen esetben a PGP termékéét, miközben az bár távolról sem tökéletes, mégis a legjobbak az esélyei, hogy a nagy hatékonyságú titkosítást a felhasználók sokaságához eljuttassa.

A világnak jó titkosítási módszerekre van szüksége, különösen olyan jó minőségű titkosítási eszközökre, amelyeket idő- és emberi erőforrástakarékos grafikus felülettel láttak el. Senki sem húz hasznot a PGP Desktop piacának megüresedéséből, ez azonban leginkább a kötelező érvényű választási lehetőségek hiánya miatt alakult így.

Egyik program sem kívánja semmilyen módon kisebbíteni ***Werner Koch*** és a GnuPG-csapat csodálatra méltó munkáját, akiket lapunk korábbi számaiban már elárasztottam szívből jövő elismerésemmel. A GnuPG hihetetlenül rövid idő alatt megbízható és érett alkalmazássá fejlődött, és máris elfoglalta az őt megillető helyet az olyan létfontosságú eszközök körében, amelyek minden Linux-változatban szerepelnek. A Linux-hívők már megszerették a GnuPG-t, barátkozz meg vele te is! Sajnos grafikusfelület-központú világunkban a GnuPG különböző felületei igazi erejét még fel kell ismerni, mielőtt igazán azt remélhetnénk, hogy a mindennapi felhasználók is készek lesznek elfogadni a GnuPG-t. Amennyiben a nem szakmai felhasználók számára ezt az utat nem tudjuk biztosítani, akár el is felejthetjük a nagy hatékonyságú titkosítás eljuttatását a felhasználók tömegeihez, még akkor is, ha ingyenesen történik. A használhatóság szempontjából a GnuPG a Linuxszal a legtágabban vett értelemben osztozik – de ó jaj, itt jön az az átkozott levelezés!

Sőt, a GnuPG a PGP Desktop működési területének csak egy részét célozza meg. Miközben a GnuPG többek között pótolja a PGP Desktop elektronikus levelezési és állománytitkosítási szolgáltatásait, nem végez állományrendszer-titkosítást, amely a legjobb szolgáltatás volt a PGP Desktop programban. A PGPdisk – a PGP állományrendszert titkosító segédprogramja – a titkosítási folyamatot egyszerűvé, gyorssá és átláthatóvá tette.

Az egyetlen dolog, amit nélkülözni volt kénytelen, az a Linux-rendszerben használható ügyfélprogram. Jómagam is – mint hordozható számítógépén két operációs rendszert üzemeltető felhasználó – ezt mindig csalódást keltőnek tartottam: egy hordozható számítógépnek minden operációs rendszerben, amelyet csak képes betölteni, titkosítással kell rendelkeznie – nincs mese!

Természetesen Linux-rendszeremre telepíthetek egy visszacsatolt titkosított állományrendszert, csakhogy az operációs rendszerek közötti átjárhatóságot ez még mindig nem teszi lehetővé. Jobb egyetlen titkosított lemezrészt megosztani a két környezet között, mint két külön környezetet fenntartani.

Ez a felvetés, ha mégannyira közvetve is, e havi témánkhoz repít bennünket, ami már nem a PGP, de még csak nem is a GnuPG: ez a BestCrypt, a nyílt forrású kereskedelmi program, amely titkosított kötetek megosztását teszi lehetővé Windows- és Linux-rendszerek között – a PGPdisktől megszokott átláthatósággal, egyszerűséggel és gyorsasággal.

Áttekintés

A BestCrypt állományrendszer-titkosító segédprogram, amellyel „befogadóegységeket" (containers) hozhatsz létre, fűzhetsz a rendszeredhez, vagy kezelhetsz bármilyen más befűzött kötethez hasonlóan a számítógépeden, viszont használaton kívül titkosított állományként tárolhatod. Ez védi az érzékeny adatokat a számítógépes kalózoktól vagy bárki mástól, aki illetéktelenül férne hozzá rendszereden tárolt adataidhoz. Minthogy a BestCrypt befogadóegységek tulajdonképpen közönséges állományok, cserélhető adathordozókon lehet őket tárolni, archiválhatók, elektronikus levélmellékletként elküldhetők – egy szó, mint száz, bármilyen más állományhoz hasonlóan kezelhetők. A BestCrypt befogadóegységeket megoszthatjuk másokkal, és távoli ügyfelek befűzhetik. Természetesen egy adott befogadóegységet rendszeréhez egyszerre csak egyetlen ügyfél fűzhet be. Ezenkívül a szóban forgó befogadóegységet mind a windowsos, mind a linuxos BestCrypt-változat képes befűzni, az egyes irányokban bármilyen működésbeli korlátozás nélkül. Mindkét változat ugyanazt az állományformát használja.

A BestCrypt beszerzése és telepítése

A BestCryptet a finnországi székhelyű Jetico, Inc. cég webhelyéről, a ➲ http://www.jetico.com/download.htm címről lehet letölteni. A webhely gyors, a BestCrypt tömör és jól összefogott – a program linuxos változata mindössze 160 K-nyi.

A Windows-változatok kissé nagyobbak, s ez kétségkívül azért van, mert bináris változatokról van szó, ugyanakkor a linuxos változatot forráskód formájában terjesztik. Írásunkban főként a linuxos változatot fogom bemutatni, azonban a windowsos változatról is ejtek pár szót.

Mielőtt megpróbálod telepíteni a BestCryptet, győződj meg róla, hogy Linux-rendszered magjának a forráskódját a */usr/src/linux* könyvtárba telepítetted-e, amelyben a */usr/src/linux* közvetett hivatkozás, vagy éppen rendszermagod főkönyvtára. Ha Linux-változatod szabványos rendszermagját használod, nincs más tennivalód, csak telepítsd a megfelelő változatszámú csomagot – ellenőrizd, hogy a változatszám megegyezik-e rendszeredével, valamint azt, hogy a */usr/src/linux* valóban a forrás főkönyvtárra mutat-e. Ha még sohasem fordítottál rendszermagot a rendszerben, a */usr/src/linux* könyvtárban az alábbi parancsokat szükséges végrehajtani:

```
make mrproper
make menuconfig  # rendszermag-forrÆsk d
                 # itt igØny szerint
                 # beÆll thatod a
                 # rendszermagot
make dep
```

Valójában még a rendszermag újrafordítása is szükségtelen – hacsak nem akarod mindenképpen elvégezni –, ehhez használhatod a `make bzImage modules modules_install` parancsot; a lényeg csupán az, hogy a rendszermag forráskódjának függőségei úgy épüljenek fel, hogy a BestCrypt forráskódjának fordításakor a kiegészítő rendszermagmodulok is helyesen illeszkedjenek a többi rendszeralkotóhoz.
A BestCryptet első alkalommal SuSE 7.1-es rendszerrel működő hordozható számítógépre telepítettem, viszont elfelejtettem, hogy azon a gépen még sohasem fordítottam rendszermagot, emiatt a BestCrypt fordítása meghiúsult. Azonban a fenti módszert lépésről lépésre végigkövetve erőfeszítéseim végül sikerrel jártak.

Rendszergazdaként használni: előnyös vagy veszélyes?

Attól függően, hogy a rendszer biztonsági jellemzői a telepítéskor vagy például a Bastille Linuxban a telepítést követően hogyan lettek beállítva, előfordulhat, hogy hozzá kell szoknunk: bizonyos fájlrendszert érintő feladatokat csak rendszergazdaként lehet elvégezni. Ez többfelhasználós rendszerben teljesen szokványos, mert a hagyományoknak megfelelően a felhasználónak nem kell tudnia fájlrendszereket formázni vagy új köteteket létrehozni.
A BestCryptet azonban nemcsak a rendszergazdáknak szánták, hanem mindenféle rendű és rangú felhasználónak. Mindazonáltal tény, hogy nem kizárólag a rendszergazda rendelkezik érzékeny adatokkal. Az eddig elmondottakon túl azonban a józan ész is azt sugallja, hogy a mindennapi tevékenységek és a nem felügyeleti jellegű feladatok elvégzése során kerüljük a rendszergazdai azonosító használatát. A szövegszerkesztési feladatainkat védelmező titkosított kötetek rendszerbe fűzése és használata nem képez, és nem is kellene, hogy felügyeleti szolgáltatást képezzen.
A BestCrypt az alapértelmezett telepítésnek megfelelően felhasználók által is üzemeltethető. Ugyanakkor a rendszer saját `mkfs` eszközeit használja az új befogadóegységek formázására, emiatt minden BestCrypt-befogadóegységet létrehozni kívánó felhasználónak végrehajtási jogosultságokkal kell rendelkeznie a */sbin/mkfs*, */sbin/mkfs.msdos* és más helyek felett.
Abban az esetben, ha gépünkön – mondjuk egy hordozható számítógépen – mi vagyunk az egyetlen felhasználók, semmi gond sincs azzal, hogy ezek az állományok a világon bárhonnan futtathatók, hiszen jellegüknél fogva talán máris ilyenek.
Ha az állományrendszerek létrehozását nem kívánjuk minden felhasználó számára lehetővé tenni, akkor azokat a bináris állományokat, amelyekhez ezek az állományok tartoznak, tegyük egy adott csoport által végrehajthatóvá és jelöljük ki a csoportot alkotó felhasználókat. Jó megoldás lehet, ha erre a célra önálló csoportot hozunk létre.
A BestCrypt a valóságban a befogadóegységeket közvetlenül kezeli, függetlenül attól, hogy a */sbin/mount* engedélyek hogyan vannak beállítva. A felhasználó csak olyan pontokra fűzhet be befogadóegységeket, amelyek felett megfelelő jogokkal rendelkezik, így felesleges aggódni amiatt, hogy egy rendszergazdai jogokkal bíró felhasználó a */bin*-en keresztül esetleg képes lenne hozzáférni a védett adatokhoz. Mi több, a BestCrypt kifejezetten támogatja titkosított saját könyvtárak létrehozását: az erre vonatkozó részletek a ➲ http://www.jetico.com/linux.htm#tricks címen olvashatók.

A BestCrypt telepítése RPM forráskódból

Ha a rendszermag forráskódja már a helyére került, és a függőségek is helyesen épültek fel, hozzáfoghatunk a BestCrypt összeépítéséhez és telepítéséhez. Ha RPM-alapú Linux-változatot használsz, töltsd le a Világhálóról az .RPM kiterjesztésű forráscsomagot – ez a cikk megírásakor `BestCrypt-1.05b-5.src.rpm` volt –, és a `-rebuild` kapcsoló használatával végezd el a program összeépítését:

```
rpm --rebuild ./BestCrypt-1.0b-5.src.rpm
```

Ez a parancs el fogja készíteni a BestCrypt bináris csomagját – Red Hat rendszeren a */usr/src/redhat/RPMS/i386* nevű könyvtárban, illetve a */usr/src/packages/RPMS/i386* könyvtárban a SuSE Linux, és talán a többi rendszer alatt is. Ezután a program telepítését már bármilyen más csomaghoz hasonlóan végezhetjük:

```
rpm -Uvh /usr/src/packages/RPMS/i386/
    ➥BestCrypt-1.0b-5.i386.rpm
```

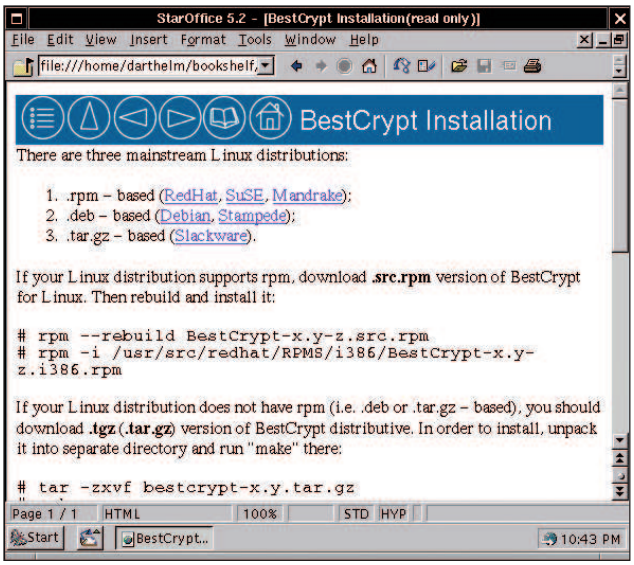
Miután minden BestCrypt bináris és `README` állomány a helyére került, a telepítés utáni héjprogram be fogja tölteni a BestCrypt rendszermagmoduljait. Ha mindezzel megvagyunk, a BestCrypt készen áll a használatra.

BestCrypt telepítése .tar állományból

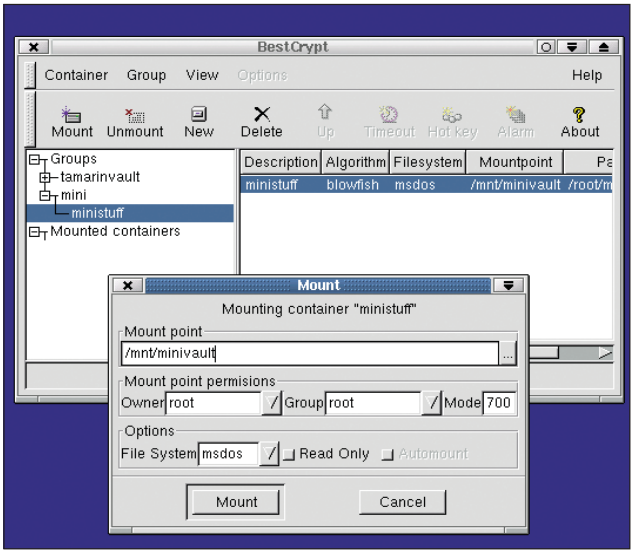
Amennyiben nem RPM-alapú Linux-változatot használsz, amilyen például a Debian vagy a Slackware, akkor az *.RPM* kiterjesztésű forráskód helyett a `tar`-állományt töltsd le – ez a cikk írásakor a `BestCrypt-1.0b-5.tar.gz` állomány volt. Csomagold ki a */usr/src* könyvtárban, tedd a */usr/src/brcypt*-et a pillanatnyi munkakönyvtárrá, és add ki a `make && make install` parancsokat. Amennyiben a rendszermag forráskódja helyesen lett telepítve, akkor a BestCrypt fordításának és telepítésének hibátlanul kell végbemennie.
A *.tar* állományban lévő `Makefile` korántsem olyan bonyolult, mint az RPM-csomag telepítő héjprogramjai. Ebben az esetben a BestCrypt első üzemszerű indítása előtt a program beállítómoduljait kézzel kell betölteni. Ennek egyszerűbb módja a BestCrypt indító héjprogramjának elindítása: `/etc/init.d/bcrypt start`

A BestCrypt leírása és vezérlőpultja

A BestCrypt programcsomagon kívül a leírást tartalmazó *.tar* csomagot is le kell tölteni, amely HTML-oldalakat tartalmazó könyvtárat rejt magában (*1. kép*).
A fentieken kívül még szükséges lehet a *BC_Panel*, azaz *BestCrypt Control Panel* (Vezérlőpult) letöltése. Ez kizárólag rpm-formátumban hozzáférhető, Debian rendszerben pedig az `alien` program segítségével telepíthető. A *BC_Panel* a BestCrypt számára grafikus felhasználói felületet biztosít, amely külsejében nagyon emlékeztet a BestCrypt windowsos változatának grafikus felhasználói felületére.
Minthogy a *BC_Panel*-t e cikk írásának idején a 0.2-1 változatszámmal látták el, valamint az a tény, hogy a parancssoros változat által nyújtott szolgáltatásokat sok tekintetben nem támogatja, arra hívják fel a figyelmet, hogy a programfejlesztési



1. kép A BestCrypt leírása



2. kép A vezérlőpult: BC_Panel v0.2b, a BestCrypt linuxos grafikus felhasználói felülete

szakasz még nem zárult le. Mindent egybevetve azonban a program számos hasznos szolgáltatást nyújt és üzembiztosnak tűnik.

A BestCrypt használata Linux-rendszerben

Egy BestCrypt befogadóegység alkalmazása nem bonyolult feladat, vessünk csak egy pillantást az alábbi példára:

```
bctool new myvault.jbc -s 150M -a twofish -d
"my test vault"
 Enter password:
 Verify password:
```

A `bctool` a BestCrypt egyetlen parancssori eszköze. Egy befogadóegység létrehozásához a `bctool` parancsnak meg kell adni a `new` (új befogadóegység) kulcsszót, az egységet jelző állomány nevét, méretét, a titkosítási algoritmust és az egység leírását. A BestCrypt ezt követően kéri a jelszót.
Használjunk könnyen megjegyezhető, de nehezen kitalálható jelszót. Annak ellenére, hogy a BestCrypt által valamennyi algoritmus támogatott – a DES-t kivéve, 128-bites vagy annál hosszabb titkosítási kulcsot használ a befogadóegységek titkosítására, és már magát az egység kulcsát is a megadott jelszóval titkosítja.
A könnyen kitalálható jelszó tulajdonképpen könnyen megfejthető objektumot jelent, függetlenül attól, hogy milyen hosszú kulccsal titkosították.
Ne felejtsük el a jelszót feljegyezni és biztonságos helyre elzárni, vagy válasszunk olyan jelszót, amelyet minden kétséget kizáróan nem fogunk elfelejteni. A Jetico cég szerint a megadott jelszavak teljes mértékben visszafejthetetlenek, helyreállíthatatlanok, és a BestCryptben semmilyen rejtett megoldás nem létezik a jelszavak korábbi titkosításának megszüntetésére. Ez alapvetően megnyugtató tény, másrészről viszont azt jelenti, hogyha jelszavunkat elfelejtjük vagy elvesztjük, adataink is visszavonhatatlanul veszendőbe mennek! Így az adatok megszerzésére törekvő számítógépes kalóz csak a találgatásban, meg a nyers erőt alkalmazó programok használatában bízhat.
A befogadóegység elkészítése után az állományrendszert is létre kell benne hozni – ez a `bctool format` paranccsal végezhető el:

```
bctool format -t msdos ./myvault.jbc
```

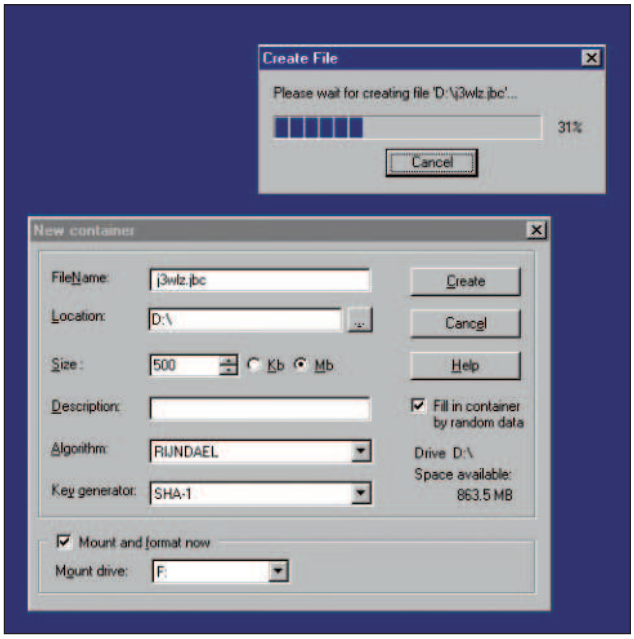
A `-t` kapcsolóval meghatározhatjuk operációs rendszerünk állományformátumát. Amennyiben ezt a befogadóegységet a BestCrypt windowsos változatával szeretnénk megosztani, típusként `msdos`-t kell megadnunk, még akkor is, ha a három karakteresnél hosszabb fájlkiterjesztést megengedő `vfat`-et, vagy a Windows 95-ben megszokott hosszú fájlneveket szeretnénk használni. Ekkor a befogadóegységet `msdos`-ként kell megformázni, de befűzésénél a `vfat` megjelölést kell használni. A BestCrypt a befogadóegység formázásakor az operációs rendszerünk által támogatott összes fájlrendszertípus használatát megengedi.
A BestCrypt befogadóegység létrehozását és formázását követően az egység befűzhető. Az ehhez szükséges `mount` utasítás nagyon hasonló a megszokott `mount` parancshoz:

```
bctool mount -t vfat ./myvault.jbc
➥./mnt/kraunj00lz
```

Innentől kezdve – egészen a leválasztásáig – a kötet ugyanúgy hozzáférhető lesz, mint bármelyik másik könyvtár. Az alapértelmezés szerint a felhasználó- és csoportváltozók annak a felhasználónak a jellemzőit veszik fel, aki a befűzést elvégezte; az engedélyeket a 0700 bájtsor jelzi, azaz `d rw- --- ---`.
Más szóval ez azt jelenti, hogy a rendszergazdától eltérő azonosítóval belépő felhasználóknak nem lesz hozzáférési joguk az egységhez, hacsak szándékosan nem más tulajdonosi jogosultságokkal és engedélyekkel végezzük el a befűzést. Természetesen a `bctool -o`, `-g` és `-m` kapcsolóival a sajátjától eltérő felhasználói azonosítót, csoportot és használati módot is kijelölhetünk. A bővebb részletek és további példák végett olvassuk át a `bctool` leírását. Amint tennivalóinkat a BestCrypt befogadóegységgel befejeztük, az egység az alább bemutatott módon kifűzhető:

```
bctool umount ./mnt/ksraunj00lz
```

Az alatt az idő alatt, amíg a BestCrypt befogadóegység nincs a rendszerbe befűzve, mentést lehet róla készíteni, vissza lehet



3. kép Új BestCrypt-befogadóegység (container) létrehozása Windows operációs rendszerben

tölteni egy korábbi állapotot, másolható vagy a többi állományhoz hasonlóan kezelhető. Ha azonban valamelyik felhasználóhoz be van fűzve, a `bctool` programon kívül mással nem módosítható vagy kezelhető.

BC_Panel: a BestCrypt linuxos grafikus felhasználói felülete

Mint fentebb már említettem, a Linux-rendszerre készített BestCrypt is rendelkezik grafikus felhasználói felülettel, e cikk írásakor azonban még csak próbaállapotú volt.
A ***BC_Panel*** (*2. kép*) a Világhálón csakis rpm-formátumban férhető hozzá. Egyrészről a ***BC_Panel*** – a bináris állománynak valójában a `bestcrypt` nevet adták, úgyhogy az egyértelműség kedvéért a továbbiakban a ***BC_Panel*** elnevezéshez fogok ragaszkodni – megbízhatóan működő, láthatóan jól megírt program, és megjelenésében nagyon is hasonlít windowsos testvérére.
A ***BC_Panel*** azonban a `bctool` szolgáltatásainak csak a töredékét tudhatja magáénak, emiatt a windowsos grafikus felhasználói felület csupán részeszköze a BestCryptnek – a másik két program fejlettségi szintjével egyáltalán nincs összhangban. Vegyük például a ***New*** (Új befogadó objektum) lehetőséget, ahol a párbeszédablak a mintegy tíz választható algoritmus-lehetőség közül csak néhányat sorol fel. Az igazi kellemetlenség azonban az, hogy a ***BC_Panel*** által felajánlott algoritmusok segítségével a befogadóegység létrehozása hibaüzenettel ér véget, hacsak nem rendszergazdaként jelentkeztünk be, és elszántuk magunkat az X-felület futtatására.
Nos, amennyiben tényleg rendszergazdaként léptünk be a rendszerbe, a BestCrypt befogadóegységeken végzett létrehozási, formázási, befűzési és leválasztási műveletek és a más algoritmussal végzett újratitkosítás egyaránt sikeresen ér véget, sőt, a BestCrypt ekkor rugalmasan képes érzékelni és felsorolni a `bctool` által befűzött egységeket, vagyis azokat a köteteket, amelyeken a felhasználó ***BC_Panelje*** olvasási engedéllyel rendelkezik.
Ezek ismeretében az ipari termelésben nem támaszkodnék e termék alkalmazására, a ***BC_Panel*** bizonyos karbantartási feladatok ellátására tűnik hasznosnak, feltéve, hogy nem törődünk azzal, hogy az X-felületet a feladat elvégzése során mindvégig rendszergazdaként kell a rendszerben használnunk. A program tevékenysége viszont mégiscsak ígéretes, és remélem, hogy a Jetico rövidesen megjelenteti a termék ipari környezetben is megbízhatóan használható változatát.

BestCrypt a Windows-rendszerben

Rendben, megállapítottam, hogy a BestCryptet könnyű Linux-rendszerben telepíteni és használni. De vajon milyen mértékig képes együttműködni a Windowszal? Milyen merevlemez-kötet-titkosítási lehetőséget tartogat a BestCrypt a Windows-felhasználók számára?
A hír mind a Windows-felhasználók, mind a Linux-hívők számára kedvező. Az elmúlt héten hordozható számítógépemen felváltva használtam Windows 98-at és SuSE Linux 7.1-et, ugyanazt a BestCrypt-egységet alkalmazva. Ez befogadóegység egy DOS/VFAT lemezterületen található, amely az írási tevékenységemhez használt operációs rendszerek munkakönyvtáraként szolgált. A BestCrypt végig hibátlanul működött, leszámítva azt a néhány ártatlan kék képernyőt, amelyet a Windows bezárásakor láttam, ugyanis a Windows gyakran arra panaszkodott, hogy a BestCrypt-kötet kifűzése után egy vagy több állomány nyitva maradt.
Adatvesztés nem fordult elő, és BestCrypt-kötet használatakor a lemez működésében sem érzékeltem lassulást. Továbbá semmiféle összeférhetetlenséget sem tapasztaltam a befogadóegységet kezelő két BestCrypt-változat között.
Mindkét program egyformán elegáns, és minden más eszköztől eltérően, amelyről mostanában írtam, gyakorlatilag nem

A program előnyei

- a program számos itt felsorolt algoritmust támogat: Blowfish, Twofish, Rijndael, IDEA, GOST, CAST, RC6, GOST, 3DES, DES. Ezenkívül ez a választék a rendszermagmodulok révén számos továbbival bővíthető
- nyílt forrású, modulrendszerű kiépítés – bárki készíthet új algoritmikus modulokat
- a Linux-változathoz tartozó forráskód – kipróbálási és tudományos célok érdekében – nyilvános
- gyors és egyszerű összeépítési és telepítési eljárás
- a jól ismert `mount` parancsot követő felépítés
- a titkosított egységeket a BestCrypt windowsos és linuxos változatai egyaránt képesek használni
- kifogástalan windowsos grafikus felület; a program linuxos grafikus felülete is hasonló elrendezésű
- a kereskedelmi termék a mérsékelt árkategóriába tartozik
- a Windows Corporate Edition (vállalati változat) tartalmazza a több távoli gépre telepített program központi kezelését végző programot
- a kereskedelmi felhasználásokhoz biztosított műszaki támogatás színvonala jó

A termék árnyoldalai

- bizonyos fokig lefordított rendszermagforrást igényel – ez elrettentheti a még kevéssé gyakorlott felhasználókat
- a linuxos grafikus felület nem támogatja az összes parancssori lehetőséget, és a windowsos grafikus felület által nyújtott lehetőségek közül sem mindet

kellett időt fektetnem a leírás hosszas böngészésébe vagy a levelezőlistákban bogarásznom, hogy a BestCryptet Windows alatt is beüzemeljem. Hogy milyen rendkívül egyszerű a BestCrypt windowsos grafikus felhasználói felületét használni, a *3. kép* mutatja be.
Már korábban megismerkedtem a nyilvános kulcsú titkosítás rejtelmeivel, és évekig más eszközöket is használtam, mint amilyen például a PGPdisk. Más szóval azt mondhatom, hogy végfelhasználóként megbízólevelem a gyanakvás – és még nagyon finoman fogalmaztam. A használhatóságot tekintve még mindig örömmel jelenthetem ki, hogy a BestCryptnek a PGPdiskkel egyenlő esélyei vannak arra, hogy a titkosítás élmezőnyébe tartozó eszközzé váljon, és a felhasználói tömegeket a kötettitkosítás és a biztonság zenszerű állapotába juttassa.
Amiben már kevésbé vagyok biztos – a fáradságos kódelemzés és titkosítás alapján –, az az, hogy a BestCrypt verhetetlen. Emberek, remélem, senki sem ragad ki részleteket a fenti mondatomból! Még szerencse, hogy a Jetico munkatársai magas elvi alapokon állnak. Minthogy sem hivatásos titkosítási szakértő, sem programozó nem vagyok, másokra kell hagynom, hogy a BestCrypt által nyújtott biztonság erejét megítéljék.

A BestCrypt nyilvánvaló biztonsága

Annyit elmondhatok, hogy a BestCrypt a közismerten jó algoritmusok lenyűgözően nagy hányadát támogatja – ahogyan talán egyes cinikusabb elmék megfogalmaznák: „crypto-kulcs-szó-megfelelő”, ideértve az Egyesült Államok kormánya által bejelentett Advanced Encryption Standard (Fejlett Titkosítási Szabvány), a Rijndael és az AES-verseny két ígéretes indulóját, ***Ron Rivest*** RC6 és ***Bruce Schneier*** Twofish algoritmusát. Ha az itt felsorolt három módszer túlságosan szokatlan lenne, a BestCrypt támogatja a 3DES-t, a több különböző kulcsmérettel a Blowfisht, az IDEA-t, a CAST-ot és az orosz szövetségi algoritmust a GOST-ot. A BestCrypt ugyan az egyszeres DES-algoritmust is támogatja, de használata nem javasolt, mivel a nyers erőt (brute force) alkalmazó programok számára a kis kulcsméret miatt könnyen feltörhető.
A Windows-felhasználók számára két további lehetőség adott: a csereállomány titkosítása, amely védelmet biztosít a jelszavak és más érzékeny adatok Windows csereállományból történő ellopása ellen, és a BCWipe, az alacsony szintű állománymegsemmisítő. E kettő közül a csereállomány-titkosító szolgáltatás még nem jelent meg linuxos változatban.
A BCWipe programot a Linux-változathoz külön kell megvásárolni, vagyis a Windows-változattól eltérően nem tartozik a BestCrypt csomagba. A BCWipe, a PGP Wipe szolgáltatásához hasonlóan a törléskor hátramaradó adatokat ismételten felülírja, ily módon téve lehetetlenné a helyreállítást szinte bármiféle lemezhelyreállító program számára, talán a legkörmönfontabbakat leszámítva, már ha azok képesek egyáltalán tenni valamit.
Megítélésem szerint a BestCrypt által nyújtott biztonság műszaki szempontból erősnek látszik: számos titkosítási és nem titkosítással kapcsolatos biztonsági módszert támogat.

Összegzés

A termék értékelése során a program által nyújtott szolgáltatásokra, a használhatóságra, a Linux-barát jellemzőkre, az általam kedvelt és hitelesnek elfogadott algoritmusok támogatására, és természetesen a küllemre összpontosítottam. Véleményem szerint a BestCrypt minden területen méltó az elismerésre, és a Jetico által képviselt magas színvonal alapján nem habozok kijelenteni, hogy titkosítási eljárásaik megvalósítása minden részletre kiterjed, és jól működik. Összességében a BestCrypt lenyűgöző programtermék. Ha a megbízhatóságból, a népszerű titkosítási algoritmusok széles kínálatának átfogó és modulrendszerű támogatásából, az általános tömörségből következtetni lehet valamire, hát akkor az az lesz, hogy nagyon biztonságos.
Lelkesedéssel ajánlhatom állományrendszerbeli szükségletek kielégítésére, különösen akkor, ha számítógépünkön Windowst is, Linuxot is használunk. E program segített felélesztени a titkosítás iránt fogékonyabb társadalomba vetett hitemet, ugyanakkor a programmal való játszadozás nagy élvezetet is jelentett számomra.

Linux Journal 2002. június, 98. szám

Mick Bauer (mick@visi.com)
hálózati biztonsággal foglalkozó szaktanácsadó. 1995 óta a Linux elkötelezett híve, 1997 óta pedig OpenBSD prófétaként tevékenykedik. Mick minden kérdést és megjegyzést szívesen fogad.

Adatok

A bemutatótermék készítője: Jetico, Inc.
A cég hivatalos honlapjának címe
➲ http://www.jetico.com

A termékek kereskedelmi ára

- Nem üzletszerű használat esetén a termék ingyenes.

A Linux-változatokhoz való programok árai

- A BestCrypt egyfelhasználós Linux-változata kereskedelmi, illetve kormányzati felhasználás esetén: 49,95 dollár.
- A BCWipe lemezterület biztonságos törlését végző segédprogram: 29,95 dollár.
- A BestCrypt egyfelhasználós változata Windows 9x/ME/NT/2K/XP rendszerekre: 89,95 dollár.
- A BestCrypt Corporate Edition Windows 9x/ME/NT/2K/XP rendszerekben használható vállalati változatának kereskedelmi, illetve kormányzati felhasználása egy felhasználóra vonatkoztatva: 149,95 dollár.

A program működésének feltételei

- Linux 2.2.0-s rendszermag, vagy ennél fejlettebb – ideértve a 2.4.x rendszermagokat is.
- A rendszermag forráskódja a */usr/local/linux* könyvtárba legyen telepítve.
- Aa telepítésre kiszemelt számítógép i386 felépítésű legyen – a program más felületen is üzembe helyezhető, de egyéb területen még nem vizsgálták a működését, és előfordulhat, hogy az adott i386-ostól eltérő környezetet a program nem támogatja.

Kagylók, héjak, miegymás...

Sorozatunk eddigi részeiben a kezdő Linux-felhasználókat ismertettük meg a legalapvetőbb műveletekkel. Eljött az idő, hogy haladóbb témákat is érintsünk. Vizsgálódásunk mostani tárgyát a parancsfájlok fogják alkotni.

Emlékezzünk csak vissza, mit is nevezünk héjnak (shell). A héj egyszerű program, tehát nem tartozik közvetlenül az operációs rendszerhez, mégis fontos szerepet tölt be az életében. Tulajdonképpen a héj nem más, mint egy olyan alkalmazás amely a felhasználó és az operációs rendszer között a kapcsolatot tartja, azaz a begépelt parancsokat értelmezi és hajtja végre. A Unix világában számos népszerű héj létezik. A legtöbb rendszerben megengedett, hogy a felhasználó saját maga választhassa meg, melyik héj alatt szeretne dolgozni. Linux alatt leginkább a bash terjedt el, mi is ezzel foglalkozunk.

A héj mint programnyelv

A héj lehetőséget ad rá, hogy úgynevezett parancsfájlokat (shell script) írhassunk, amelyek segítségével bizonyos feladatokat önműködővé tehetünk.
A Unix parancsfájljait gyakran szokták a DOS világából jól ismert batch fájlokhoz hasonlítgatni, ám ez nem feltétlenül szerencsés, ugyanis ezek a parancsállományok sokkal szélesebb felhasználásnak örvendenek. Akár programokként is felfoghatnánk őket. Cikkünk további részében e nyelv bemutatásával foglalkozunk.
A parancsfájlt közvetlenül maga a héjprogram értelmezi és hajtja végre. Minden parancsfájlnak a `#!/bin/bash` bűvös sorralal kell kezdődnie. Ezzel azt mondhatjuk meg, hogy az adott fájlban leírtakat melyik alkalmazásnak kell átadni, ami majd remélhetőleg értelmezi és végrehajtja őket.
Hozzunk létre a `cat` segítségével egy egyszerű parancsfájlt (a `^D` helyén nyomjuk le a Ctrl+D kombinációt):

```
ras:~$ cat > sellscript
#!/bin/bash
echo "Szia vilÆg!"
^D
```

Mielőtt végrehajtanánk, ne felejtsük el a parancsfájlt futtathatóvá tenni:

```
ras:~$ chmod a+x shellscript
```

A héj változók használatára is lehetőséget teremt:

```
#!/bin/bash
VALTOZO="Hell   vilÆg!"
echo $VALTOZO
```

Az előző sorokban a változó értékadására láthatunk példát. Ha a változóhoz rendelt értéket szeretnénk használni, a változó neve elé egy `$` (dollárjelet) kell írnunk. Léteznek úgynevezett beépített héjváltozók is. Ezek értéke alapértelmezés szerint lett meghatározva, és számos közülük csak olvasható, azaz értékeit nem változtathatjuk meg. Vegyük sorra röviden a legfontosabb héjváltozókat!
A `$HOME` a felhasználó saját könyvtárát tartalmazza. A `$PATH` a keresési útvonalakat (ez határozza meg, hogy a héj hol keresse a végrehajtandó programokat), a `$PS1` pedig a parancsjelet (prompt) határozza meg. A `$LOGNAME` a felhasználó bejelentkezési azonosítóját tárolja, a `$TERM`-ben pedig a használt terminál típusát találhatjuk.
A változók harmadik típusát az úgynevezett kapcsolók képezik. Ezeket tulajdonképpen a héj meghívásakor a parancssorban kell megadni. A kapcsolókhoz a *$sorszÆm* segítségével férhetünk hozzá. Ha például a parancsfájlt a `./shellscript alma korte afonya` sorral hívjuk meg, akkor a `$1` változónak ***alma***, a `$2`-nek ***korte*** a `$3`-nak pedig ***afonya*** lesz az értéke. Ezeket egyébként helyhezkötött változóknak is nevezzük.
A tulajdonságok könnyebb megértése végett lássunk egy egyszerű példát:

```
#!/bin/bash
echo $3 $2 $1
```

A parancsfájlt a következő módon futtassuk le:

```
ras:~$./shellscript egy 2 harom
```

Láthatjuk, hogy végeredményül a megadott kapcsolókat kapjuk vissza, csak éppen fordított sorrendben: ***harom 2 egy***.

Ezzel a módszerrel összesen kilenc kapcsolót kezelhetünk, a 0. kapcsoló maga a parancsfájl neve.
A `$*` az értékként megadott összes kapcsolót tartalmazza. A `$#`-ban pedig a megadott kapcsolók számát találhatjuk.
Valószínű, hogy az `echo` parancs rendeltetése nem igényel részletes magyarázatot, ám érdemes kitérnünk néhány egyedi esetre. Megfigyelhettük, hogy minden egyes `echo` parancs esetén a szöveg kiírásán kívül egy új sor karakter is a kimenetre került, azaz a következő `echo` utasítás alkalmával a megadott szöveg már egy új sorba íródik ki. Ezt a `\c` kifejezéssel szüntethetjük meg:

```
ras:~$ echo-e"Szia vilÆg! \c"
```

A `-e` kapcsolóval arra utasítjuk az `echo` parancsot, hogy értelmezze a fordított perjellel kezdődő vezérlőjeleket. Ha a kiírandó szövegben különleges karaktereket is használni szeretnénk, szintén a `\` előtag segít nekünk. Például:

```
ras:~$ echo "A mai Ærfolyam
➥szerint \$1 = 260 forint".
```

Mint már említettük, parancsfájljainkban elágazásokat és ciklusokat is használhatunk. A továbbiakban ezeket taglaljuk egy kicsit részletesebben.
Kezdjük az elágazásokkal! Itt a következő szerkezetet kell használnunk:

```
if parancs1
   then parancs2
   fi
```

Ez azt jelenti, hogyha az első parancs sikeres (azaz 0 visszatérési értéket ad vissza), akkor hajtódik végre a ***parancs2***.
A szerkezetet a `fi` kulcsszó zárja. Nézzünk egy egyszerű példát:

```
if date | grep "Jan   1"
   then echo "Boldog œjØvet!"
fi.
```

Ebben az esetben a `date` utasítás (amely

a pillanatnyi dátumot írja ki) kimenetét átadjuk a grep parancsnak, amely megnézi, hogy a kapcsolónak megadott karaktersorozat szerepel-e a szövegben. Ha igen, egyrészt kiírja azokat a sorokat, amelyekben szerepel, másrészt a visszatérési értéke igaz, azaz 0 lesz. Ha nem, 0-tól különböző visszatérési értéket kapunk. Ám mi a helyzet akkor, ha azt szeretnénk, hogy a parancsfájl akkor is tegyen valamit, ha a parancs1 hamis? Jelen esetben a dátum nem január elseje. Ebben az esetben az else kulcsszót kell használnunk:

```
if date | grep "Jan  1"
  then echo "Boldog œjØvet!"
  else echo "Az œjØvre
  ➥sajnos mØg vÆrnunk kell."
  fi
```

Ha bonyolultabb elágazást szeretnénk gyártani, a helyzetet leegyszerűsítheti az elif kulcsszó alkalmazása, amely az else és az if:

```
if date | grep "Jan  1";
 then echo "Boldog œjØvet!"
 elif date | grep "Dec 25"
   then echo "LegalÆbb
        ➥karÆcsony van..."
   else echo "Se karÆcsony,
        ➥se œjØv :-("
fi
```

Ha karakterláncokat, illetve szám- (numerikus) értékeket szeretnénk egymással összehasonlíthatni, a test utasítást kell használnunk. A test úgynevezett külső parancs, azaz nem közvetlen része a héjnak, mégis kifejezetten a parancsfájlokhoz találták ki. Az utasítás kapcsolójául egy kifejezést kell adnunk. Ha ez a kifejezés igaz, a visszatérési érték 0, egyébként valami más, azaz hamis lesz. A kifejezés a következőképpen nézhet ki: N mßvelet M. N és M a parancsfájlunk két változója, a művelet pedig azt határozza meg, hogy N-nek és M-nek milyen viszonyban kell lenniük egymással, hogy a kifejezés értéke igaz legyen. Ha számértékekkel dolgozunk, a következő műveleteket használhatjuk:

-eq: N és M egyenlő;
-ne: N és M nem egyenlő;
-gt: N nagyobb, mint M;
-ge: N nagyobb vagy egyenlő, mint M;
-lt: N kisebb, mint M;
-le: N kisebb vagy egyenlő, mint M.

Nézzünk meg egy egyszerű példát:

```
if test $# -lt 1
   then echo "A parancsfÆjl-
   ➥nak nem adtÆl ØrtØket"
   fi
```

Ha karakterláncokkal dolgozunk, általában a következő két műveletre lesz szükségünk: = (egyenlőségjelre), amennyiben a két karaktersorozat egyenlő, illetve !=, ekkor a két karaktersorozat nem egyenlő. Ezenkívül használhatjuk még a -z-t, amely akkor lesz igaz, ha a megadott karakterlánc hossza 0, illetve ennek fordítottját, a -n-t (amikor a hossz nagyobb, mint 0).
A test utasításnak van még egy hasznos szolgáltatása, mégpedig a fájltípusok ellenőrzése. Ebben az esetben a kifejezésnél a mßvelet fÆjlnØv írásmódot kell alkalmaznunk. A műveletek a következők lehetnek:

-s: létezik-e a megadott állomány, és ha igen, nem üres-e;
-d: a megadott állomány könyvtár-e;
-r: a megadott állomány nem könyvtár-e;
-w: a megadott állomány írható-e;
-r: a megadott állomány olvasható-e.

Nézzünk egy példát erre is:

```
if test ! -s $1
   then echo "A kapcsol kØnt
   ➥megadott fÆjl nem
   ➥lØtezik vagy  res!"
   fi
```

A ! (felkiáltójel) a kifejezés tagadására szolgál. A bash egyébként felkínál egy egyszerűbb módszert a test használatához: elég ha a kifejezést szögletes zárójelbe tesszük: [-s proba.txt].

Most térjünk át a ciklusokra! Az első az úgynevezett while ciklus, szerkezete a következő:

```
while parancs1
   do parancs2
   done
```

A parancs2 több parancsot is tartalmazhat, amely addig kerül végrehajtásra, amíg a parancs1 visszatérési értéke igaz. A while-hoz nagyon hasonló az until ciklus, a különbség csak annyi, hogy a while kulcsszó helyett until-t kell használnunk, és a ciklus addig fut, amíg a parancs1 nullától különböző értékkel tér vissza (tehát a ciklusfeltétel a while-nál használt feltétel fordítottja). Példaként nézzünk meg egy olyan parancsfájlt, amely nem csinál mást, mint a kapcsolóként megadott összes állomány tartalmát kiírja a képernyőre. Ehhez szükségünk lesz a shift utasításra is, segítségével a helyhezkötött változókat léptethetjük ($(N+1) → $1, $(N+2) →$2, stb.). Az N a shift utasításnak megadott kapcsoló, ha nem adjuk meg, akkor értékként önműködően 1-et vesz fel.

```
while test $# -gt 0; do
  echo "Az Ælomæny neve: $1"
  echo "Az Ællomæny tartalma:"
  cat $1
  shift
  done
```

A fenti példát a for számlálóciklus segítségével egyszerűbben is megírhatjuk. Ez annyiban különbözik az előbb említett feltételes ciklusoktól, hogy a ciklusmagban felsorolt utasítások végrehajtásának száma nem egy feltételtől, hanem egy előre meghatározott értéktől függ:

```
for i
  do
  echo "Az Ælomæny neve: $i"
  echo "Az Ællomæny tartalma:"
  cat $i
done
```

A for szerkezete a következő:

```
for i in wordlist ; do
  parancs(ok)
  done
```

A wordlist nem más, mint egy karakterláncokból álló felsorolás. Az i mindig az éppen sorra kerülő elemet veszi fel a wordlist-ből, majd a héj lefuttatja a ciklusmagot. Ha nem adunk meg wordlist-et, annak szerepét a $* tölti be. Az első esetben tehát az i értéke $1 lesz, a másodikban $2 és így tovább.

Előfordulhat, hogy a ciklus futását meg akarjuk szakítani. Erre három lehetőségünk van: a continue parancs hatására az adott ciklus végére ugrik (új kört kezd), a break eredménye, hogy a ciklusszerkezet után folytatódik a feldolgozás, az exit eredménye pedig a parancsfájl futásának azonnali befejezése.

Garzó András
(garzoand@interware.hu) Körülbelül három éve foglalkozik Linux- és más Unix-rendszerekkel. Legjobban az operációs rendszerek lelkivilága érdekli, de nyitott egyéniség. Kedvenc étele a palacsinta, és van egy Richard nevű macskája. Minden észrevételt, megjegyzést, levelet szívesen fogad.

Futtassunk mindent Linux alatt!

Most induló sorozatunkban a Linux x86 alatti emulációt taglaljuk a különféle számítógép-felületekhez.

Remélem, mindenki emlékszik még a PC előtti gépekre, amilyen a ZX Spectrum, a HT1080 Z, a C-64, a Primo és társaik voltak. Nos, ezekre a gépekre nagyon sok program készült – kedvenc játékom a Bruce Lee volt, amit sohasem tudtam végigjátszani, mivel körülbelül a harmincadik pályánál mindig kifagyott, mindig ugyanott és mindig ugyanúgy. Egyszer azt gondoltam, Linux alatt is kipróbálom, kísérletet is tettem a végigjátszására, mivel akkor még úgy gondoltam, biztosan a lemezemen lévő fájl sérült. Az új fájlt egy magazin CD-mellékletén találtam. Nagy izgalommal nekikezdtem a környezet létrehozásának, elindítottam a játékot, eljutottam ugyanahhoz a pályához, na, gondoltam, most ugrik a majom a vízbe, és igen, kedves olvasó, ugyanúgy kifagyott a program, mint a jó öreg C-64-en tette ezt tíz évvel ezelőtt, viszont a nosztalgia és az ebből adódó édesbús érzés kárpótolt a csalódásért.

Ez a rövidke ízelítő, azt hiszem, előrevetíti a cikksorozat első néhány részének a tartalmát, a Linux alatti számítógép-emuláció azonban nemcsak ezekre az öreg gépekre terjed ki, hanem PalmOS, Macintosh, HP számológép-környezet is létezik. Ezeket szintén szándékunkban áll bemutatni.

A cikkek során a Debian/GNU SID operációs rendszert használom majd, így meglehet, hogy a különböző Linux-változatokban nem minden fog ugyanúgy működni, mint ahogyan én leírom. De egy kis utánjárással biztosan bárki próbálkozását siker koronázza. Ez a téma számos feladat megoldását és komoly fejtörést igényel a programok használójától, de még inkább a programok készítőjétől, mivel nem egyszerű feladat Linux alá lefordítani egy teljesen más felépítésű gép rendszerhívásait, tehát mindenképpen magas szintű programozói tudás kell. Ahogy nőtt a gépek bonyolultsága, úgy vált a feladat bonyolultsága is egyre nagyobb mértékűvé. Erre a legjobb példa talán a Basilisk II 68 K Macintosh-emulátor. Nagyon bonyolult és összetett feladatok végrehajtására is alkalmas. Mondanom sem kell, hogy ezekre a Macintosh-gépekre nagyszámú jó minőségű program készült, így a segítségükkel ezeket is használni tudjuk.

A Debian/GNU SID alatt kiadott `apt-cache search emulator` parancs a következő listát adja (kiválogatva és lefordítva):

bochs – x86 PC-emulátor
tkisem – Graphical SPARC-emulátor
wine – Windows-emulátor
apple2 – Apple-emulátor
atari800 – Atari-emulátor
dosemu – DOS-emulátor
gnuboy-sdl – Game Boy-emulátor
gtktiemu – Texas Instruments számológép-emulátor
nestra – Nintendo Entertainment System-emulátor
pose – PalmOS-emulátor
spectemu-x11 – ZX Spectrum-emulátor X11-hez
uae – Amiga-emulátor
vice – Commodore-emulátor

xapple2 – Apple-emulátor
xtrs – emulátor a TRS-80 Model I/III/4/4P számítógépekhez
zsnes – Super Nintendo Entertainment System (TM) emulátor
dgen – Sega Genesis/MegaDrive-emulátor
ines – NES/Famicom/Dandy game system-emulátor
simh – különféle öreg számítógépek emulátora
snes9x – Super NES-emulátor
spim – MIPS R2000/R3000-emulátor
stella – Atari 2600-emulátor X11-hez
xzx – X11-alapú ZX Spectrum-emulátor

Miként a fenti listából is kitűnik, számtalanféle környezet programjaihoz találunk segítséget – ezért sorozatunk a legismertebb és egyben a leghasznosabb programok ismertetését tűzi ki célul. Várjuk kedves olvasóink kéréseit és ötleteit, esetleg tapasztalatait is a cikksorozattal kapcsolatban.

Csontos Gyula

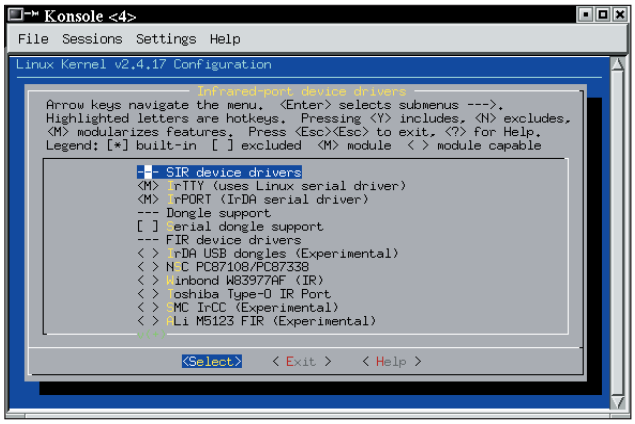
Kapcsolódó címek

- http://www.linuxlinks.com/Software/Emulators/
- http://www.kearney.net/~mhoffman/basiliskII/system753_tutorial/linux/
- http://bochs.sourceforge.net/
- http://www.dosemu.org/
- http://www.winehq.com/
- http://www.plex86.org/
- http://www.classicgaming.com/cb64/cb64unix.shtml
- http://www.uni-mainz.de/~bauec002/FRMain.html
- http://www.cs.cmu.edu/~dsladic/vice/vice.html

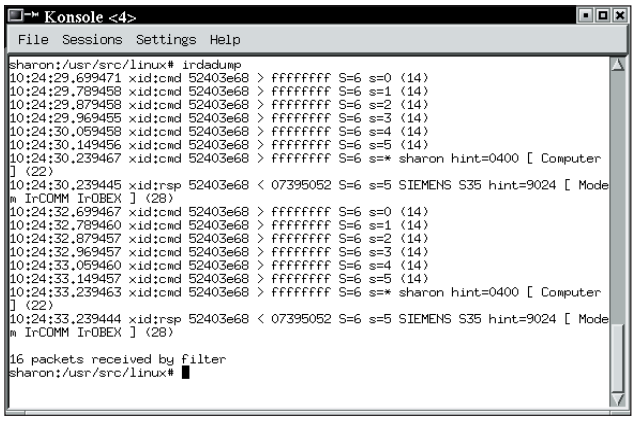
A Vörösök Linux alatt is hódítanak

Természetesen nem kell mindenáron politikai tartalmat és elfogultságot feltételezni, ennél sokkal egyszerűbb megoldása van a fenti címnek. Ennek a cikknek a témája hordozható számítógépek bekötése az Internet szövevényes hálózatába mobiltelefon segítségével. Szinte minden hordozható számítógép rendelkezik infravörös kapuval, ennek és az ugyan nem olcsó, viszonylag lassú, de bárhol használható mobiltelefonos internetkapcsolatnak a beállítását mutatom be. Ahhoz, hogy a fentebb említett kapcsolatot létre tudjuk hozni, az alábbiakra lesz szükségünk:

- egy linuxos gépre infravörös kapuval,
- egy infrakapus mobiltelefonra,
- a pppconfig beállítóprogramra és
- infravörös kapcsolatot támogató programokra.



1. kép A rendszermag beállítása



2. kép Az irdadump program kimenete

Első lépésként győződjünk meg az infravörös eszközök meglétéről: a rendszermagnak az `irda` és az `irtty` modulokkal kell rendelkeznie. Töltsük be őket a memóriába, és ha ezek a modulok mégsem léteznének, egy rendszermagfordítással kiküszöbölhetjük a hiányosságot. Az *1. képen* a saját rendszerem infravörös beállításait láthatjuk. Ha olyan infrakapuval rendelkeztek, mely megfelel a FIR szabványnak és a képen látható lapkakészletek közül való, a gyorsabb FIR adatátvitelt is használhatjátok. Miután ezzel megvagyunk, ellenőrizzük a kapunkat. Helyesen beállított infrás telefonunkat helyezzük a kapuhoz – a telefonok beállításai egyéniek, ezért erre most nem térek ki, saját Siemens S35-ös telefonomon az infrakaput és a FAX-/adatvétel lehetőségeket kellett bekapcsolnom a ***Beállítások/ FAX/adat mód*** menüpontban. Győződjünk meg a segédprogramok meglétéről (`irdadump`, `irdaping`), ha nem találhatók, a hiányt Debian/GNU Linux alatt az ***irda-common*** és az ***irda-tools*** csomagok telepítésével szüntethetjük meg.
Ha ezzel is végeztünk, indítsuk el az `irdadump` programot. Ez a program a két infravörös eszköz között zajló teljes kommunikációt kiírja, ezért elég sok adat jelenhet meg a képernyőn – ám ne ijedjünk meg tőlük. Keressünk egy olyan sort, ahol a rádiótelefonunkra jellemző leírást találhatjuk, ez az én esetemben a `Siemens S35` karaktersorozat megjelenése (lásd a *2. képet*).
Ha idáig eljutottunk, akkor elmondhatjuk, hogy a kapcsolat a telefon és a számítógép között működik. Következő lépésként internetkapcsolatot kell létrehoznunk.

Kapcsolat létrehozása

Telefonos internetkapcsolat létrehozására én a `pppconfig` programot szoktam használni, mivel különféle kapcsolatainkat egyszerű felületen állíthatjuk be. Mobilszolgáltatónknál internet-előfizetéssel kell rendelkeznünk, ami teljes mértékben olyan, mint a vonalas internet-előfizetés – ez például a westel900-nál havidíjmentes, csak az interneten eltöltött időt kell kifizetni.
Ha még nem rendelkezünk ilyennel, keressük fel mobilszolgáltatónkat, és kérjünk felhasználónevet, valamint jelszót a kapcsolathoz. Szükség lesz még a felhívandó telefonszámra, a DNS-kiszolgáló(k)ra, az infravörös kapura (ez az én gépemen a */dev/ircom0*), a kapu sebességére és a kapcsolat azonosító eljárására (ez a PAP lesz).
Természetesen bármelyik jobban kedvelt beállítóprogramot is használhatjuk erre a célra.
Ha a kapcsolatot sikeresen létrehoztuk, próbáljuk is ki mindjárt. Ezt Debian rendszeren a `pon kapcsolatneve` paranccsal tehetjük meg a legkönnyebben.
Ez a kapcsolat ugyan nem túl gyors, de nyugodtan levelezhetünk vele!

Jó internetezést bárhol!

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

Kapcsolódó címek

- http://www.pasta.cs.uit.no/mailman/listinfo/linux-irda
- http://www.linux-embedded.com/howto/IR-HOWTO.html
- http://www.tldp.org/HOWTO/Infrared-HOWTO/index.html

A PCI Hot Plug eszközillesztő fájlrendszer

Greg ismerteti, hogyan valósít meg a PCI Hot Plug-mag egy RAM-alapú fájlrendszert, és hogyan teheted meg ugyanezt a saját illesztőiddel te is.

Múlt év május 14-én ***H. Peter Anvin*** az alábbiakat tette közzé a „linux-kernel" levelezési listán: „***Linus Torvalds*** ideiglenes szünetet kért az új eszközök azonosítóinak kiosztásában. Reményei szerint ennek köszönhetően egy új és jobb eszközelnevezési rendszer fog kialakulni."
Peter tulajdonképpen a linuxos nevek és azonosítók kiosztásáért felelős „hivatal", így bárkinek, aki rendszermag-illesztőprogram fejlesztésére adja a fejét, hozzá kell fordulnia egy fő- és egy alazonosítóért, amit az illesztőprogramjához kap. Az új azonosítók kiosztásának befagyasztása természetesen parázs vitát gerjesztett: vajon milyen módszerrel lehetne hatékonyabban kezelni az eszközök neveit, azonosítóit? Az egyik ötlet az volt, hogy egy illesztőprogramot kellene készíteni, amely egy a felhasználó és az illesztőprogam közti párbeszédet kezelő fájlrendszert valósítana meg.
Ez idő tájt én éppen a Compaq által a saját kiszolgálóihoz írt PCI Hot Plug illesztőprogramot csinosítgattam. A PCI Hot Plug illesztőprogram lehetővé teszi a PCI-kártyák üzem közben történő kikapcsolását, kivételét a gépből, egy másik kártya behelyezését, majd a foglalat és az új kártya visszakapcsolását – feltéve, hogy megfelelő vassal rendelkezünk. Különösen akkor hasznos az ilyesmi, ha olyan kiszolgálót üzemeltetünk, amelynél a leállás megengedhetetlen, azonban újabb hálózati kártyák telepítésére, a meghibásodottak cseréjére vagy egyéb karbantartási munkák elvégzésére szükség lehet.
A PCI Hot Plug illesztőprogramot eredetileg arra tervezték, hogy karakteres eszközként tartsa a kapcsolatot a felhasználói réteggel. Az `ioctl` hívásokat az eszközcsomóponthoz kellett intézni, így lehetett a PCI-foglalatokat és a foglalatok visszajelző fényeit leállítani és bekapcsolni, valamint az eszközökön különféle hibaellenőrző programokat futtatni. Annak érdekében, hogy a rendszerben lévő PCI-foglalatok számát és állapotát (energiaellátás és állapotjelzők) le lehessen kérdezni, egy csak olvasható */**proc*** könyvtárfa is elérhető volt.
Miközben azon dolgoztam, hogy a PCI Hot Plug központi részeit kiszedjem a Compaq-illesztőprogramból, hogy a felhasználó felé más PCI Hot Plug illesztőprogramok is közös felülettel rendelkezhessenek, rájöttem, hogy egyetlen fájlrendszerrel jobban meg lehetne oldani mind a PCI-foglalatokkal kapcsolatos adatok, mind a felhasználótól érkező beavatkozások kezelését. Az összes, az illesztőprogrammal kapcsolatos adatot és beavatkozási műveletet egyetlen helyről lehetne kezelni, és nem kellene két különböző jellegű felületet fenntartani.
A PCI Hot Plug illesztőprogram a 2.4.15-ös változattól kezdve a fő rendszermagfa részét képezi, és ***pcihpfs*** néven fájlrendszert tesz elérhetővé – az illesztőprogram ennek segítségével kezelhető. Amikor befűzzük a fájlrendszert, egy 3, 4, 5 stb. nevű könyvtárakból álló fát kapunk, melyben a számok a PCI-foglalatok fizikai számát jelzik. Az egyes foglalatokhoz tartozó könyvtárakban található fájlokból olvashatjuk ki a foglalattal kapcsolatos adatokat. A ***power*** és ***attention*** nevű fájlok írhatók is, így `0/1` értéket megadva a tápellátás és a jelzőfény ki- és bekapcsolható. A ***test*** fájlt hardveres tesztparancsok küldésére használhatjuk. Az ***adapter*** állományból tudhatjuk meg, hogy van-e kártya a foglalatban, a ***latch*** fájl pedig a foglalathoz – fizikailag – tartozó retesz állapotáról tájékoztat, feltéve, hogy a gépben található ilyen.
Ha tehát például az 5-ös foglalatot akarjuk bekapcsolni, az alábbi parancsot kell kiadnunk a ***pcihpfs*** főkönyvtárában:

```
echo 1 > 5/power
```

Ha a foglalatban kártya van, a teljes PCI-indítási folyamat le fog futni, amelynek részeként PCI-adatokkal a */**sbin**/**hotplug***-ot is meghívja, így a rendszer az eszközhöz tartozó modult önműködően betöltheti.
A fájlrendszernek köszönhetően a felhasználói programoknak karakteres eszközhöz nem kell különleges `ioctl()` hívásokat intézniük, és számos lehetőség közül választhatnak, hogyan is kívánják eszközeiket vezérelni.
A cikk további részei arról szólnak, hogy a PCI Hot Plug-mag mi módon valósít meg egy RAM-alapú fájlrendszert, és miként teheted meg ugyanezt a saját illesztőiddel te is.
Először is az illesztőprogramban be kell vezetni a fájlrendszert (declaration). Ehhez a `DECLARE_FSTYPE` makrót lehet használni, amely az ***include**/**linux**/**file.h*** állományban található.
A ***pci_hotplug*** illesztőprogram a `DECLARE_FSTYPE` makrót az alábbiak szerint használja:

```
static DECLARE_FSTYPE(pcihpfs_fs_type,
➥"pcihpfs", pcihpfs_read_super,
➥FS_SINGLE | FS_LITTER);
```

Ezzel a `struct pcihpfs_fs_type` típusból egy `file_system_type` nevű statikus példány jön létre, valamint megtörténik a struktúra néhány mezőjének kezdeti értékadása is. A ***name**/**név*** mező értéke most `pcihpfs`, ezt fogják a felhasználók a fájlrendszer befűzésére használni, így nem árt mindig olyan nevet választani, aminek értelme is van, és amelyet más fájlrendszer nem használ a rendszermagban. Az `FS_SINGLE` és `FS_LITTER` jelzőket is állítsuk be.
Az `FS_SINGLE` azt jelenti, hogy fájlrendszerünkben a szuperblokkból csak egyetlen példány lesz. Emiatt bárhova is fűzzük be a fájlrendszert, a fájlrendszerben minden befűzési pont ugyanarra helyre fog mutatni (ne feledd, ugyanazt a fájlrendszert a könyvtárfa különböző pontjain is befűzheted).
Az `FS_LITTER` beállítással megadtuk, hogy a könyvtárfát a `dcache`-ben szeretnénk tartani. A beállítást azért használjuk, mert fájlrendszerünk teljes egészében a memóriában foglal majd helyet, nem lesz lemez, háttértár vagy egyéb fizikai adattároló hozzárendelve.
A `pcihpfs_fs_type` `read_super` mezője arra a függvényre mutat, amelyet akkor hívunk meg, amikor a rendszermag a fájlrendszer szuperblokkját akarja beolvasni. A szuperblokk egy fájlrendszernek az a szerkezete, amely a teljes fájlrendszert magát írja le. A rendszermag ezt a függvényt akkor hívja meg,

amikor valaki be akarja fűzni a fájlrendszert. A függvény meghívásakor a rendszermag tudomására kell hoznunk, hogy a fájlrendszer pontosan hogyan épül fel.
Mielőtt azonban befűzhetnénk a fájlrendszert, meg kell mondanunk a rendszermagnak, hogy maga a fájlrendszer elérhető, létezik. Ezt egy egyszerű `register_filesystem()` hívással tehetjük meg, amelynek egyetlen átadott értéke a `file_system_type`. A műveletet a `pci_hotplug` modul kezdő értékeit megadó függvényével végezhetjük el, az alábbi kódrészlettel:

```
dbg("fÆjlrendszer bejegyzØse.\n");
result = register_filesystem
➥(&pcihpfs_fs_type);
if (result) {
   err("a register_filesystem hibÆval tØrt
   ➥vissza, hibak d: %d\n", result);
  goto exit;
 }
```

A `pci_hotplug` modul leállításakor fájlrendszerünk bejegyzését az alábbi kódrészlettel hasonló módon töröljük:

```
unregister_filesystem(&pcihpfs_fs_type);
```

Miután bejegyeztük a fájlrendszert, néhány virtuális fájlt szeretnénk létrehozni, amelyek révén a felhasználók írhatják és olvashatják az illesztőprogram által módosított és exportált értékeket. Ha egy felhasználó azelőtt fűzi be a fájlrendszert, hogy fájlt próbálna létrehozni, a rendszermag már rendelkezni fog a fájlrendszer egy virtuális megtestesülésével. Előfordul, hogy a fájlrendszer még nincs befűzve, létrehozása után azonban – mielőtt új fájlt hozhatnánk létre – a rendszermagnak be kell fűznie (egyéb esetben az állomány létrehozása sikertelen lesz). Ennek elhárítására két megoldás kínálkozik. Az egyiknél addig várunk, amíg a fájlrendszer befűzése ténylegesen megtörténik (a befűzésről a `read_super` függvény meghívása tudósít bennünket), és csak ezután hozzuk létre az új állományokat. Ennél a módszernél a befűzésnél elég sokat kell dolgoznunk, és folyamatosan figyelemmel kell kísérnünk, hogy a fájlrendszer éppen be van-e fűzve. Ne feledjük ugyanis, hogy különböző időpontokban kell fájlokat hozzáadnunk vagy törölnünk. Az ***usbdevfs*** fájlrendszer (ennek nincs köze a ***devfs***-hez, csak szerencsétlen módon hasonlít a nevük) kiváló példa az olyan fájlrendszerre, amelyik ezt a megoldást alkalmazza.
Most azonban nem kívánjuk folyamatosan követni, hogy fájlrendszerünk be van-e fűzve vagy sem, egyszerűen csak fájlokat akarunk létrehozni és törölni, amikor éppen kedvünk tartja. A második megoldás szerint a rendszermagot arra kell utasítanunk, hogy a fájlrendszert belsőleg fűzze be. Ezzel elhárul a befűzési állapot követésének gondja. Az *1. listán* ez a megoldás látható. Rágjuk át egy kicsit ezt a kódrészletet, és próbáljuk megérteni, mit és hogyan tesz. Arra az esetre nézvést is kiváló példát adunk, ha megfelelő zárolási módszereket kell kidolgozni, mert például a rendszermag többprocesszoros gépen fut.
Az alábbi sorral elsőként egy `spin`-zárat kérünk `mount_lock` néven:

```
spin_lock(&mount_lock);
```

Ezt a zárat belső számlálónk védelmére használjuk abban az esetben, ha a fájlrendszer már be lenne fűzve. Igen, igaz, korábban azt mondtam, hogy a befűzési állapotot nem szeretnénk nyomon követni. Semmi baj nincs, ezzel az egyszerű függvénnyel, valamint a fájlrendszer leválasztására használttal (ezt később ismertetem) sokkal könnyebb dolgozni, és könnyebb őket megérteni is, mint a befűzési állapot folyamatos követésére használt módszert. Elég belenézni a 2.4.18-as és korábbi rendszermagok ***drivers/usb/inode.c*** állományába, és máris láthatjuk, mi kell ez utóbbi megfelelő működéséhez.
Miután a `spin`-zár megvan, ellenőriznünk kell, hogy a belső `mount` változó be van-e állítva:

```
if (pcihpfs_mount) {
        mntget(pcihpfs_mount);
        mntget(pcihpfs_mount);
        spin_unlock (&mount_lock);
        goto go_ahead;
 }
```

Ha igen, az `mntget()` meghívásával növeljük belső befűzési számlálónkat. Az `mntget()` egy egyszerű, helyben kifejtett (inline) függvény az ***include/linux/mount.h*** állományban.

1. lista Kódrészlet, melyben a rendszermagot a fájlrendszer belső befűzésére utasítjuk

```
 static int get_mount (void)
{
    struct vfsmount *mnt;

    spin_lock (&mount_lock);
    if (pcihpfs_mount) {
        mntget(pcihpfs_mount);
        ++pcihpfs_mount_count;
        spin_unlock (&mount_lock);
        goto go_ahead;
    }

    spin_unlock (&mount_lock);
     mnt = kern_mount (&pcihpfs_fs_type);
    if (IS_ERR(mnt)) {
        err ("nem siker lt befßzni a
        ➥fÆjlrendszert  kilØpØs!\n");
        return -ENODEV;
    }
    spin_lock (&mount_lock);
    if (!pcihpfs_mount) {
        pcihpfs_mount = mnt;
        ++pcihpfs_mount_count;
        spin_unlock (&mount_lock);
        goto go_ahead;
    }
    mntget(pcihpfs_mount);
    ++pcihpfs_mount_count;
    spin_unlock (&mount_lock);
    mntput(mnt);

go_ahead:
    dbg("pcihpfs_mount_count = %d\n",
       ➥pcihpfs_mount_count);
    return 0;
}
```

Ezután belső számlálónkat növeljük, eleresztjük a spin-zárat, és a függvény végére ugrunk, ugyanis végeztünk (néha még a rendszermagba is belefér egy-egy goto).
Egyéb esetben a fájlrendszer nincs befűzve. Eleresztjük tehát a spin-zárat:

```
spin_unlock (&mount_lock);
```

majd a kern_mount meghívásával belsőleg fűzzük be a fájlrendszert:

```
mnt = kern_mount (&pcihpfs_fs_type);
if (IS_ERR(mnt)) {
    err ("nem siker lt befßzni a fÆjlrendszert...
        ➥kilØpek!\n");
    return -ENODEV;
 }
```

A spin-zárat eleresztjük, mivel a kern_mount() függvény futása eltarthat egy ideig, és az is előfordulhat, hogy a rendszermag más folyamatnak adja át az erőforrásokat. Ügyeljünk arra, hogy spin-zárat nem szabad fenntartani, ha a schedule() meghívására sor kerülhet – ilyenkor ugyanis elég csúnya dolgok történhetnek…
Most, hogy a fájlrendszert befűztük, újra elkérjük spin-zárat:

```
spin_unlock (&mount_lock);
```

de egyúttal ellenőrizzük, hogy a belső, befűzést jelző számlálónk továbbra is nulla-e:

```
if (!pcihpfs_mount) {
        pcihpfs mount = mnt;
        pcihpfs_mount_count;
        spin_unlock (&mount_lock);
        goto go_ahead;
 }
```

„Állj!" – mondhatnád ekkor. „Mit akarunk a pcihpfs_mount-tal? Tudjuk, hogy nulla az értéke, alig pár kódsorral előbb vizsgáltuk meg! Miért bántjuk újra?" Ne feledd, hogy a kern_mount() meghívásánál – mint szó volt róla – a vezérlés más folyamathoz is kerülhet! Ha a kern_mount() hívásakor ez történik, és egy másik folyamat is ugyanezt a kódrészletet hajtja végre (miért is ne, hiszen több processzorunk van, és egy időben természetesen több felhasználói szál is futhat), előfordulhat, hogy a fájlrendszert sikeresen befűzte, és a pcihpfs_mount változó értékét növelte. Na, ezért kell újra elvégeznünk az értékvizsgálatot.
Ha nem volt másik folyamat, amely közbelépett, és a fájlrendszert befűzte volna, a fájlrendszer mutatóját későbbi műveletekhez mentjük, belső számlálónkat megnöveljük, a zárat eleresztjük és kilépünk.
Ha egy másik folyamat köreinket megzavarta, és befűzte a fájlrendszert, a következőket tesszük:

```
mntget(pcihpfs_mount);
++pcihpfs_mount_count;
spin_unlock (&mount_lock);
mntput(mnt);
```

Ez azzal egyezik meg, amit hasonló helyzetben cselekedtünk a függvény elején.
A fájlrendszer leválasztására használt kódrészlet még egyszerűbb:

```
static void remove_mount (void)
{
        struct vfsmount *mnt;

        spin_lock (&mount_lock);
        mnt = pcihpfs_mount;
        --pcihpfs_mount_count;
        if (!pcihpfs_mount_count)
                ➥pcihpfs_mount = NULL;

        spin_unlock (&mount_lock);
        mntput(mnt);
        dbg("pcihpfs_mount_count = %d\n",
            ➥pcihpfs_mount_count);
 }
```

Ebben a függvényben is szert teszünk egy zárra (a fájlrendszer befűzésekor használttal megegyezőre), a fájlrendszer befűzési alkalmainak számát nyilvántartó számlálót csökkentjük (minden befűzéshez tartoznia kell egy leválasztásnak), majd eleresztjük a zárat. Ezután szólunk a rendszermagnak, hogy a fájlrendszert le akarjuk választani – ehhez már csak az mntput() meghívása szükséges.
Amikor a rendszermag lényegében azért akarja befűzni a fájlrendszert, mert meghívtuk a kern_mount() függvényt – vagy mert egy felhasználó kezdeményezte –, a pcihpfs_read_super() függvény kerül meghívásra. Ebben létre kell hoznunk néhány rendszermagszerkezetet, amelyek megadják a fájlrendszer jellemzőit, valamint a fájlrendszer élettartama alatt a rendszermag által meghívott függvények fellelhetőségét. Mindezt az alábbi néhány kódsorral érhetjük el:

```
sb->s_blocksize = PAGE_CACHE_SIZE;
sb->s_blocksize_bits = PAGE_CACHE_SHIFT;
sb->s_magic = PCIHPFS_MAGIC;
sb->s_op = &pcihpfs_ops;
```

Először is megadjuk, hogy fájlrendszerünk blokkmérete megegyezik a lapozó gyorsítótár méretével, közöljük a fájlrendszer magic-azonosítóját, amelynek a rendszerben egyedinek kell lennie, majd megadjuk super_operations szerkezeti függvényeink listájának elérhetőségét.
A szuperblokk központi fájlleírójának (root inode) kezdeti értékeit az alábbiakkal adjuk meg:

```
inode = pcihpfs_get_inode(sb, S_IFDIR | 0755, 0);
if (!inode) {
       dbg("%s: az i-node nem Ørhetı
       ➥el!\n",__FUNCTION__);
       return NULL;
 }
```

A pcihpfs_get_inode() működéséről még lesz szó, ha viszont a fentiek sikeresek, a fő dentry-t hozzárendeljük a most létrehozott fájlleíróhoz, majd mentjük a dentry-t a szuperblokkszerkezetbe:

```
root = d_alloc_root(inode);
if (!root) {
       dbg("%s: a gy kØr dentry nem Ørhetı
       ➥el!\n",
           __FUNCTION__);
```

2. lista Új fájlleíró létrehozása

```
static struct inode *pcihpfs_get_inode
(struct super_block *sb, int mode, int dev)
{
    struct inode *inode = new_inode(sb);

    if (inode) {
      inode->i_mode = mode;
      inode->i_uid = current->fsuid;
      inode->i_gid = current->fsgid;
      inode->i_blksize = PAGE_CACHE_SIZE;
      inode->i_blocks = 0;
      inode->i_rdev = NODEV;
      inode->i_mapping->a_ops =
      ➥&pcihpfs_aops;
       inode->i_atime = inode->i_mtime =
       ➥inode->i_ctime = CURRENT_TIME;
      switch (mode & S_IFMT) {
      default:
        init_special_inode(inode, mode, dev);
        break;
      case S_IFREG:
        inode->i_fop =
        ➥&default_file_operations;
        break;
      case S_IFDIR:
         inode->i_op =
         ➥&pcihpfs_dir_inode_operations;
         inode->i_fop =
         ➥&pcihpfs_dir_operations;
         break;
      }
    }
    return inode;
}
```

```
      iput(inode);

      return NULL;
}
sb->s_root = root;
return sb;
```

Ez minden, amire a szuperblokk kezdeti értékadásához szükség van – ezzel a rendszermag a fájlrendszert sikeresen befűzte.
A `pcihpfs_get_inode()` is egy olyan függvény, amelyet létre kell hoznunk a fájlrendszerhez. Akkor hívódik meg, amikor a fájlrendszerben új fájlleírót kell létrehoznunk.
A 2. *lista* szemlélteti, miként végzi el mindezt a `pci_hotplug` illesztőprogram.
Először a rendszermag `new_inode()` függvényét kell meghívnunk, hogy új fájlleíró szerkezet jöjjön létre, és a kezdeti értékadás megtörténjen. Ha ez sikeres, több mezőt is fel kell töltenünk a szükséges adatokkal. Az ***i_uid*** és az ***i_gid*** mezőkbe a jelenlegi folyamat `uid`- és `gid`-értékei kerülnek, ily módon biztosítható a fájlleíró létrehozójának későbbi hozzáférése.
Az ***i_atime***, ***i_mtime*** és ***i_ctime*** mezők a fájlleíró hozzáférés idejét, az utolsó módosítás és az utolsó változás idejét adják meg. Ezeket a pillanatnyi időpontra állítjuk be. Ha ez a fájlleíró

5. lista Az fs_remove_file() függvény meghívása

```
static void fs_remove_file
➥(struct dentry *dentry)
{
    struct dentry *parent = dentry->d_parent;

    if (!parent || !parent->d_inode)
         return;

    down(&parent->d_inode->i_sem);
    if (pcihpfs_positive(dentry)) {
        if (dentry->d_inode) {
             if (S_ISDIR(dentry->d_inode->
             ➥i_mode))
                  vfs_rmdir(parent->
                  ➥d_inode,dentry);
             else
                  vfs_unlink(parent->
                  ➥d_inode,dentry);
        }

        dput(dentry);
    }
    up(&parent->d_inode->i_sem);
}
```

„rendes” fájltípus, akkor a `default_file_operations` halmazra mutatunk, mint azon függvények halmazára, amelyeket a fájlleíró szerephez jutásakor – megnyitás, írás, olvasás stb. – kell meghívni. Ha a fájlleíró könyvtárleíró, az alapértelmezett halmaz a könyvtárkezelő függvényekre fog mutatni. Ha a fájlleíró nem „rendes” és nem könyvtárleíró, hagyjuk, hogy a kezdeti értékadást a rendszermag az `init_special_inode()` meghívásával végezze el.
Most, hogy sikeresen elvégeztük a fájlrendszer belső befűzését, hogyan hozhatunk létre a felhasználók által is írható-olvasható fájlokat? Először az `fs_create_file()` függvényt kell meghívni, amelynek átadjuk a létrehozandó fájl nevét és kezelési módját, egy mutatót a szülőkönyvtárra – ha ez `NULL`, a fájl a fájlrendszer főkönyvtárába kerül –, egy másik mutatót a fájlhoz hozzárendelendő adathalmazzal, valamint egy a fájl elérésekor meghívott fájlműveletek halmazát megadó mutatót.
Itt hívjuk meg a `pcihpfs_create_by_name` függvényt, amellyel a megadott adatok alapján egy új `dentry` jön létre.
A `dentry` létrehozása után mentjük az adatmutatót, és a `dentry file_operations` mutatója azokra a függvényekre irányul, amelyeket a `dentry` fájlleírójának elérésekor ténylegesen meg akarunk hívni.
A `file_operations` szerkezet, amelyet egy fájlleíróhoz rendelünk, a létrehozott fájl típusától függően változik. A ***power*** állomány esetében, amely az adott PCI-foglalat ki- vagy bekapcsolt állapotát adja meg, valamint a ki- és bekapcsolásra is használható, az alábbi szerkezetet használjuk:

```
static struct file_operations
power_file_operations = {
        read:           power_read_file,
        write:          power_write_file,
        open:           default_open,
};
```

Ez esetben a `power_read_file` és a `power_write_file` függvény érdekes. Ezek kerülnek meghívásra, ha valaki a fájlt olvasni vagy írni próbálja. A további függvények akkor jutnak szerephez, ha más műveleteket végzünk az állományon. Az `open()` hívás hatására a rendszermag a `default_open`, az `llseek` hatására a `default_file_lseek()` függvényt hívja meg és így tovább.
A `power_read_file()` rendkívül egyszerű függvény – mindössze a megadott PCI-foglalat ki- vagy bekapcsolt állapotát kell visszaadnia. A vonatkozó kódrészlet:

```
page = (unsigned char *)
       ➥__get_free_page(GFP_KERNEL);
if (!page)
          return -ENOMEM;

retval = get_power_status (slot, &value);
if (retval)
         goto exit;
len = sprintf (page, "%d\n", value);
```

A kódrészlet lefoglal egy darab memóriát (egy lapnyit), lekérdezi a PCI-foglalat állapotát (a `get_power_status()` függvény meghívásával), majd az állapotot tükröző karakterláncot kiírja a lefoglalt memóriarészbe. A memóriarészt ezután átmásolja a felhasználói memóriaterületre. Ne feledjük, hogy az eredeti memóriarész a rendszermaghoz tartozik – ha a felhasználók oldaláról is elérhetővé akarjuk tenni, az alábbi kódrészletet végre kell hajtani:

```
if (copy_to_user (buf, page, len)) {
       retval = -EFAULT; goto exit;
 }
```

A részletben a `buf` mutató a felhasználói memóriarészben lévő átmeneti tárra mutat, amelyet eredetileg a `read()` hívásnak adtunk át. Amikor a felhasználó kiadja az alábbi parancsot

```
cat /tmp/pcihpfs/slot2/power
```

az eredmény a következő:

```
1
```

A `power_write_file()` függvény ugyanilyen egyszerű. Azt szeretnénk, ha a felhasználó egy egyszerű `echo` paranccsal vezérelhetné a PCI-foglalat tápellátását, például:

```
echo 1 > /tmp/pcihpfs/slot3/power
```

Ezzel a paranccsal a harmadik PCI-foglalat tápellátását lehet bekapcsolni. A művelethez az átadott értéket karakterlánc formátumból át kell alakítanunk bináris számmá, és meg kell határoznunk, hogy melyik PCI-foglalatokra jellemző egyedi függvényt kell meghívnunk (lásd a 4. listát, 36. CD Magazin/PCI könyvtár).
Először létrehozunk egy a felhasználói karakterláncnál egy bájttal nagyobb átmeneti tárat, és feltöltjük nullákkal. Második lépésként a tartalmát a felhasználói részből a rendszermag átmeneti tárába másoljuk, ezután a `simple_strtoul()` függvénnyel bináris formátumba alakítjuk, majd értékétől függően a `disable_slot()` vagy az `enable_slot()` függvényt hívjuk meg a megadott PCI-foglalathoz.

A fenti két egyszerű függvénnyel bármely felhasználó által elérhető illesztőprogram-felületet hozunk létre anélkül, hogy különleges, `ioctl`-jellegű hívásokat kellene végrehajtani.
Amikor az illesztőprogram leáll, minden olyan fájlt el kell távolítania, amelyet a fájlrendszerben hoztunk létre, így válik lehetővé a fájlrendszer leválasztása és a hozzárendelt memória felszabadítása. Ehhez az `fs_remove_file()` függvény kell meghívni (lásd az *5. listát*).
A függvénynek át kell adni egy mutatót, amely az `fs_create_file` hívás által visszaadott `dentry`-t határozza meg. Megvizsgálja, hogy a `dentry` rendelkezik-e érvényes szülővel, ugyanis egy `dentry` csak ebben az esetben távolítható el. Ezt követően a rendszermag VFS-rétegétől kéri a `dentry` eltávolítását (ekkor eltérő hívásokra kerülhet sor, attól függően, hogy a `dentry` fájlra vagy könyvtárra hivatkozik).
Leírtuk az alapvető fájlrendszerműveleteket, amelyek egy illesztőprogramban megvalósított fájlrendszer létrehozásához szükségesek. Ha pontosabb leírást szeretnél arról, hogy a különféle részek hogyan működnek együtt, vess egy pillantást a Linux-rendszermagfa ***drivers/hotplug/pci_hotplug_core.c*** állományában található programkódra.
A cikk a 2.4-es rendszermagnál szükséges tennivalókat tárgyalta. A 2.5-ös változatnál – annak köszönhetően, hogy a `ramfs` függvények túlnyomó részét exportálják – számos dolgot könnyebben meg lehet oldani. A RAM-alapú fájlrendszerek között így nagyobb mértékben oszthatók meg a programkódok, a programozók tehát kevesebb munkával is célt érhetnek, és a hibás megvalósítások készítésének valószínűsége is csökken.

Köszönetnyilvánítás

Szeretnék köszönetet mondani ***Pat Mochelnek*** a ***ddfs/driverfs*** kód megírásáért, amelyen a ***pcihpfs*** programkód jelentős része alapult. A ***driverfs*** egy új fájlrendszer a 2.5-ös rendszermagban, amelynek segítségével az illesztőprogramok készítői illesztőprogramjaik különleges adatait a felhasználói területre exportálhatják, valamint faszerkezetben az összes eszközt elérhetővé tehetik, így a tápellátást kezelő eszközök sokkal egyszerűbbekké válhatnak.
Szintén szeretném megköszönni ***Al Viro*** értékes válaszait a VFS-sel kapcsolatos kérdéseimre, amelyek révén egy fájlrendszert ilyen kevés kóddal sikerült megvalósítani.

A listák megtalálhatóak a 36. CD Magazin/PCI könyvtárában.

Linux Journal 2002. május, 97. szám

Greg Kroah-Hartman
(greg@kroah.com) jelenleg a Linux USB és a PCI Hot Plug rendszermagfelelőse. Az IBM-nél dolgozik, ahol számos, a Linux rendszermagjával kapcsolatos kérdéssel foglalkozik.

További érdekességek

Ha további tájékoztatást szeretnél kapni a PCI Hot Plug illesztőprogrammal kapcsolatban, látogass el a ➲ http://www.kroah.com/linux/hotplug címre, ahol a *pcihpfs* kezelését grafikus felületen lehetővé tévő felhasználói alkalmazásokra is találsz hivatkozásokat.

Az operációs rendszerek belügyei

Hogyan is működik egy mai korszerű operációs rendszer? Miként osztja meg gépünk erőforrásait a párhuzamosan futó alkalmazások között? Miként gazdálkodik a rendelkezésre álló memóriával, és hogyan tárolja adatainkat a merevlemezen?

Most induló sorozatunk az ehhez hasonló kérdésekre keresi a választ, leginkább csak elméleti síkon, de a gyakorlatban is bemutatjuk: vizsgálódásunk alanyául az egyik legelterjedtebb nyílt forráskódú rendszert, a Linux-rendszermagot választottuk.

Még mielőtt buzgón a mély vízbe vetnénk magunkat, tisztáznunk kell néhány alapvető fogalmat, mivel pontos ismeretük rendkívül fontos az operációs rendszerek lelkivilágának mélyebb megismeréséhez. Elképzelhető, hogy sorozatunk első részének mondanivalójával már sok olvasó tisztában van, de mi azt szeretnénk, hogy minél szélesebb olvasói réteg vehesse hasznát írásunknak. Ezért a hangsúlyt nem a tömörségre és teljességre, hanem a könnyebb megértésre próbáljuk helyezni. Teljességre egyébként is hiú ábránd lenne törekedni, sorozatunk témája ugyanis rendkívül bonyolult és összetett, vaskos könyvek és tanulmányok láttak róla napvilágot. Mi elsősorban e szövevényes téma azon szeleteivel foglalkozunk, amelyek egy átlagos felhasználót érdekelhetnek.

Az itt bemutatott elvek gyakorlatban történő megvalósítása rettenetesen nehéz és összetett feladat, részletes taglalására újságunk keretein belül nem is lenne mód, ezért megelégszünk annyival, ha elméleti síkon be tudjuk mutatni egy mai korszerű operációs rendszer működését. Akit a téma ennél is mélyebben érdekel, javasoljuk, lapozgassa az ajánlott olvasmányként feltüntetett műveket.

Akkor kezdjük az elején! Mi a feladata tulajdonképpen az operációs rendszernek? E kérdés megválaszolása nem is olyan egyszerű. Az operációs rendszernek alapvetően két különböző, ám egymással szoros kapcsolatban lévő feladatnak kell eleget tennie: az ***erőforrás-kezelésnek***, és egy olyan ***egységes programozási felület megteremtésének***, amelyen keresztül a felhasználói alkalmazások erőforrásainkat könnyebben használhatják. Nézzük meg, mit is takarnak ezek a kifejezések!

Az operációs rendszer egyrészt erőforrás-kezelő. Erőforrás alatt azokat az eszközöket (például háttértárolókat, nyomtatókat, egereket), illetve adategységeket (például egy állományt vagy egy adatbázis meghatározott rekordját) értjük, amelyet ***egy időben csak egy program érhet el***. Hogy egy időben valóban csak egy alkalmazás férhessen az adott erőforráshoz, azt az operációs rendszer biztosítja.

Ennek könnyebb megemésztése végett nézzünk egy hétköznapi esetet: ki szeretnénk nyomtani egy állomány tartalmát. Többfeladatos rendszerekben (például Linux, Windows, OS/2) bőségesen előfordulhatnak olyan helyzetek, hogy két párhuzamosan futó alkalmazás egyszerre akar egy erőforrással dolgozni, jelen esetben nyomtatni. Ha az erőforrásokhoz bárki bármikor korlátlanul hozzáférhetne, könnyen adódhatnak nagy galibák. Például míg mi buzgón állományunk tartalmát nyomtatjuk, addig valaki egy másik terminálról szintén nyomtatási parancsot adhat ki. Eredményül nagy összevisszaságot kapunk, amelyen mindkét egyszerre kinyomtatott állomány tartalmát megtalálhatjuk, egymással összeolvadva.

Könnyű belátni, mennyire kulcsfontosságú, hogy az erőforrások használata az operációs rendszer által szabályozva legyen. Ez a szabályozás a gyakorlatban úgy valósul meg, hogy a különböző erőforrások közvetlen kezeléséről maga az operációs rendszer gondoskodik. Ha ki szeretnénk nyomtatni egy állományt, meg kell kérnünk a rendszert, hogy tegye meg nekünk (egy átlagos alkalmazás a legtöbb rendszerben a párhuzamos kapuhoz nem is férhet hozzá közvetlenül). Például Unix-alapú rendszerekben ezt úgy tehetjük meg, hogy a kinyomtatni kívánt állományt elhelyezzük egy erre a célra kijelölt könyvtárban. Egy másik, vele együtt futó különleges alkalmazás – a nyomtatódémon – folyamatosan ellenőrzi e könyvtár tartalmát, és ha talál benne nyomtatásra szánt anyagot, annak tartalmát elküldi a párhuzamos kapura, majd munkája befejeztével az állományt kitörli. Ha időközben újabb kinyomtatásra váró anyag érkezik, egészen addig nem foglalkozik vele, míg előző feladatát nem teljesítette.

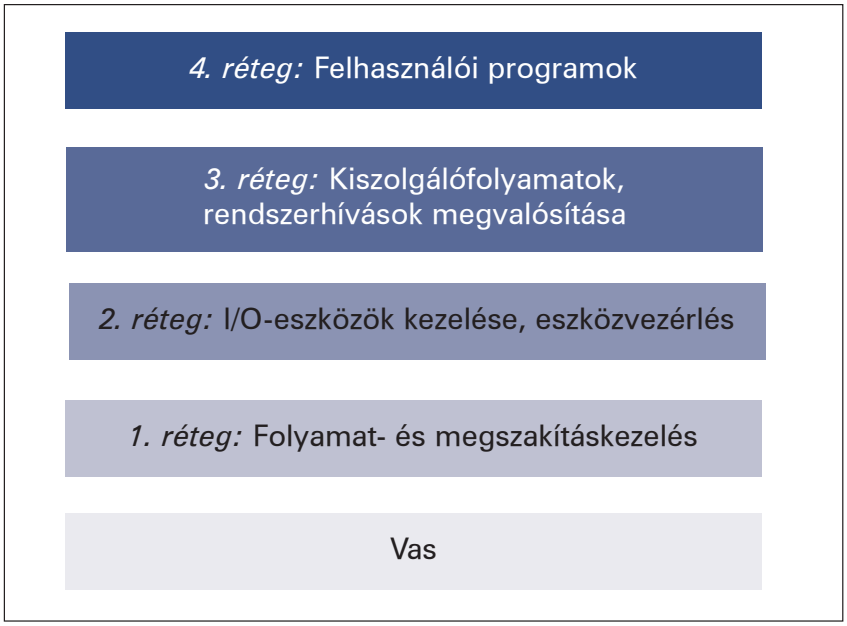
A többfeladatos operációs rendszerek lehetővé teszik, hogy több alkalmazást is futtathassunk párhuzamosan. Azt a legkisebb egységet, amely párhuzamos feldolgozásra kerülhet, ***folyamatnak*** (process) nevezzük. Tulajdonképpen minden elindított alkalmazás egy-egy folyamatnak felel meg a rendszerben.

Mint tudjuk, a processzorok soros feldolgozásúak (az utasításokat egymás után egyesével hajtják végre), azaz egy időben mindig csak egy folyamat futhat. Az operációs rendszernek tehát nemcsak az erőforrásokat kell beosztania a programok között, hanem azt is, hogy melyik folyamat mikor és mennyi ideig futhat. Amikor ez az idő letelik, a futási lehetőséget egy másik folyamat kapja meg. Szakszerűen mondva az operációs rendszernek a folyamatok ***ütemezéséről*** is gondoskodnia kell.

Jogosan merül fel a kérdés, hogy mi a helyzet a manapság egyre jobban terjedő többprocesszoros rendszerek, illetve a géptelepek (több gépből összeállított szupergéptek) esetében? Itt az operációs rendszernek egy sokkal bonyolultabb ütemező algoritmussal kell rendelkeznie, de több processzor hatékonyabb kihasználásához a felhasználói programok felőli támogatásra is szükség lesz.

Mit értünk ez alatt? Logikusnak tűnhet, hogy ahány processzort tömünk számítógépünk belsejébe, a feldolgozás annyiszor gyorsabb lesz. Ez az elképzelés azonban téves, ugyanis sebességnövekedést csak több alkalmazás párhuzamos futtatásakor észlelhetünk. Ám a hálózati kiszolgálóktól eltekintve a tapasztalat azt mutatja, hogy a felhasználók általában csak egy folyamat feldolgozásán szeretnének gyorsítani. A megoldás kézenfekvő: az adott folyamatot egyszerre több processzoron futtatjuk. Ám ezt csak akkor tehetjük meg, ha az alkalmazást eleve erre felkészülve írták meg, azaz párhuzamosították. A párhuzamos

feldolgozásra szánt programok írása egy kissé más szemléletet igényel, mint a hagyományos sor programoké. Nem ritka, hogy egy alkalmazást a párhuzamosítás érdekében szinte teljesen át kell írni. Sőt, bizonyos esetekben csupán kis százalékban érünk el sebességnövekedést, tehát programunkat minden olyan alkatrészre külön meg kell írnunk, amelyen futtatni szeretnénk.
Érthető, hogy aki felhasználói alkalmazások fejlesztésére adta a fejét, nem kíván ilyen mélységben megismerkedni az alkatrész programozás rejtelmeivel.



A Linux-rendszer felépítése

ugyanis amennyi időtöbbletet nyerünk azáltal, hogy több számítást egyszerre tudunk végezni, ugyanannyit kell áldoznunk a több processzoron való futtatás összehangolására.
De térjünk vissza eredeti témánkhoz! Az operációs rendszerek másik fő feladatköre következik: az erőforrások elérésének megkönnyítése. A számítógép alkatrészeinek programozása sosem volt kényelmes feladat. A felhasználói alkalmazások által gyakran használt legalapvetőbb műveletek (például egy fájlból való olvasás) megvalósítása is komoly kihívást jelent. Például nem mondhatjuk meg a merevlemeznek, hogy az adott fájlból olvassa ki az első száz bájtot. Mi csak szektorokat olvastathatunk be vele, de még ehhez a legegyszerűbb művelethez is rengeteg adat ismerete szükséges, mint például a pályánkénti szektorok száma, vagy hogy hány darab cilinder található a merevlemezen. Sőt, olyasmire is figyelni kell, hogy mennyi időt vesz igénybe az olvasófej állítása, illetve a motor felpörgetése (ezt az időt ugyanis programunknak várakozással kell töltenie). Nem is beszélve arról, hogy a különböző cégek által gyártott eszközöket az esetek többségében nem azonos módon kell programoznunk, Csak annyit szeretne, hogy a fájlból való olvasást egy egyszerű eljárás meghívásával elvégezhesse, és értékként csak a fájl nevét, illetve a beolvasandó bájtok számát kelljen átadnia. Az, hogy az állomány milyen módon, illetve milyen típusú háttértárolón van tárolva, a programozót a legkevésbé sem érdekli.
Ezért az operációs rendszer számítógépünk szolgáltatásait a felhasználói programok számára kibővíti. Eme bonyodalmas műveletek – mint például a fájlkezelés – megvalósítását az operációs rendszer magára vállalja, és olyan felületet hoz létre, amely a felhasználó szeme elől „eltakarja” az alkatrész programozás sötét világát. Ha úgy tetszik, egy olyan virtuális számítógépet kapunk, amelynek programozása lényegesen egyszerűbb.
Azok a szolgáltatások, amelyek a programozók számára leegyszerűsítik az erőforrások használatát, ***rendszerhívások*** formájában valósulnak meg. A rendszerhívások tulajdonképpen olyan eljárások, amelyeket a felhasználói alkalmazások bármikor szabadon meghívhatnak. Hogy miképpen az, gép- és rendszerfüggő. Például MS-DOS-ban a hexadeximális 21-es megszakítást kellett meghívnunk (a megszakításokról később bővebben beszélni fogunk), de az értékeket előtte meghatározott regiszterekben kellett tárolnunk. Unix-alapú rendszerek alatt a rendszerhívások azonban C könyvtárak formájában is elérhetőek, így a C nyelvben írt programokból közvetlenül meghívhatók.
Meg kell jegyeznünk, hogy az operációs rendszerek többsége csak rendszerhívásokon keresztül engedélyezi az erőforrások elérését, közvetlenül nem. Ugyanis csak így tarthatja meg állandó befolyását számítógépünk felett, tehát kizárólag így szavatolhatja, hogy egy erőforrást egyszerre mindig csak egy program használ, illetve ellenőrizheti, hogy egy művelet végrehajtásához a futó alkalmazásnak megvan-e a kellő jogosultsága. Ezek együttesen biztosítják, hogy semmilyen felhasználó által indított program nem lesz képes a rendszerünket összeomlasztani.
A POSIX-szabvány (amely azt írja le, hogy egy Unix-alapú rendszernek milyen feltételeknek kell megfelelnie) 53 rendszerhívást határoz meg. Ezek olyan alapvető műveleteket valósítanak meg, mint folyamatok és jelek kezelése, fájl-, könyvtár-, illetve fájlrendszerkezelés, valamint a dátum- és az idő kezelése. A Linux ezeknél természetesen sokkal több rendszerhívást tartalmaz, amelyek például a további perifériák kezelését vagy a hálózati szolgáltatásokat valósítják meg.
Mielőtt továbbmennénk, tisztáznunk kell, hogy a gépünkön található programrengetegből melyek alkotják közvetlenül operációs rendszerünk részeit, és melyek nem. Azok a programok, amelyek úgynevezett ***felhasználói módban*** futnak, semmiképp sem tartozékai az operációs rendszernek. Nemcsak a különböző felhasználói alkalmazások (például szövegszerkesztők) sorolhatók ebbe a csoportba, hanem például az úgynevezett héjprogramok is. Ezek olyan alkalmazások, amelyek a felhasználó és az operációs rendszer közötti kapcsolattartást biztosítják, segítségükkel adhatunk utasításokat a rendszernek. Ide tartoznak a parancsértelmezők (például a bash, a csh), de tulajdonképpen az ablakkezelők is. Hiába fontos tehát egy rendszer életében a héjprogram, mégsem közvetlen tartozéka az operációs rendszernek.
A rendszerprogram azon részeit hívjuk operációs rendszernek, amelyek úgynevezett rendszermag-, más néven felügyelt módban futnak. Ezek sokkal több mindent megtehetnek, mint a felhasználói szinten futtatott alkalmazások, például közvetlen beviteli/kiviteli műveleteket hajthatnak végre, illetve további

kedvezményeket is élvezhetnek (például nagyobb elsőbbséget).
Fontos megjegyezni, hogy a magmódban elhelyezkedő programokhoz és az alkatrészekhez felhasználók közvetlenül nem férhetnek hozzá. Míg tehát a felhasználó saját maga választhatja meg például azt, hogy melyik héj alatt szeretne dolgozni, vagy hogy melyik levelezőprogram alatt óhajtja a leveleit olvasgatni, addig például a billentyűzet kezeléséért felelős eljárásokat nem cserélheti csak úgy le.
Most nézzük meg egy mai korszerű operációs rendszernek a felépítését, a Linuxét. A Linux belső felépítését vizsgálva négy eltérő, ám jól meghatározott feladatot ellátó réteget különböztethetünk meg. A legalsó rétegnek két feladata van: a folyamatok ütemezése és a közöttük zajló kapcsolattartás lehetővé tétele. Az utóbbit IPC-nek (InterProcess Communication – folyamatközti kapcsolattartás) hívjuk, és a későbbiekben nagyon fontos szerepe lesz. A legalsó réteg felelős az eszközmegszakítások fogadásáért. Sorozatunk következő részében e réteg működésével bővebben is foglalkozunk.
A második réteg legfontosabb feladata az erőforrás-kezelés megvalósítása. Ebben a rétegben helyezkednek el az eszközmeghajtók (device drivers). Ezek már folyamatok, azaz a felhasználói programokkal párhuzamosan futó eljárások, de a hardveres egységekhez közvetlenül hozzáférhetnek, és az ütemező is „méltányosabban” bánik velük. Az első és a második réteg együtt alkotja a kernelt, közismertebb nevén a rendszermagot.
A harmadik rétegben a rendszerhívások végrehajtásáért felelős úgynevezett ***kiszolgálófolyamatok*** helyezkednek el, amelyek a felhasználói programoknak nyújtanak hasznos szolgáltatásokat. Külön kiszolgálófolyamat fut például a memóriakezeléssel vagy a fájlrendszerrel kapcsolatos rendszerhívások kezelésére. Fontos megjegyezni, hogy a kiszolgálók már teljesen önálló folyamatok, és a hatáskörük is erősen korlátozott (például nem hajthatnak végre közvetlen I/O-műveleteket), mégis megkülönböztetett helyzetben vannak a felhasználói folyamatokkal szemben. Továbbá a kiszolgálófolyamatok előbb indulnak el, mint bármelyik felhasználói folyamat, és egészen addig futni fognak, amíg rendszerünket le nem állítjuk. Ezért a kiszolgálófolyamatok nem felhasználói módban futnak, de már nem is közvetlenül a rendszermag részei.
A felhasználó szemszögéből nézve ez érdektelen, mivel a kiszolgálófolyamatok forrását a Linux-mag forrása tartalmazza, és fordításkor az is a rendszermaggal együtt fordul.
Az utolsó szinten a felhasználói alkalmazások foglalnak helyet, azaz itt fut a szövegszerkesztő, a héj és a különböző egyedi felhasználói folyamatok, a démonok.
A démonok az itt bemutatott kiszolgálófolyamatokkal sok hasonlóságot mutatnak, mivel a rendszer indításától általában ők is egészen a leállításig futnak, de érdemi tevékenységet csak egy meghatározott esemény bekövetkeztekor végeznek (például egy időpont eljövetelekor vagy egy hálózati kapura való kapcsolódási kérelem esetében). Ha ez megtörténik, azonnal munkába állnak, és elvégzik a kijelölt feladatot, majd ismét „álomba szenderülnek”. Ugyanakkor a démon csak egy egyszerű felhasználói folyamat, ugyanolyan megkötések vonatkoznak rá, mint például szövegszerkesztőnkre, ebben az értelemben nem tekinthető az operációs rendszer részének. Ám a Unix világában a démonoknak mégis fontos szerep jutott, ugyanis velük hajtatnak végre minden olyan műveletet, amelyet egy felhasználói szintű program is elvégezhet.
A különböző hálózati kiszolgálók (például a web-, az FTP-, a levélkiszolgálók) is démon formájában futnak.
Ne bánkódjunk, ha a GNU/Linux felépítése még mindig homályos számunkra. A következő résztől kezdve sorban végigmegyünk az összes rétegen, és részletesen, példákkal szemléltetve mutatjuk be feladatukat és működésük lényegét.
Ennyi volt, amit feltétlenül tudnunk kell ahhoz, hogy mélyebben belevessük magunkat az operációs rendszerek működésének tanulmányozásába. A legközelebbi alkalommal már érdekesebb témákkal foglalkozunk: először megismerkedünk a Linux-rendszermag forrásának felépítésével, majd megnézzük, hogy miként kezeli a folyamatokat, illetve hogyan teszi lehetővé a közöttük való kapcsolattartást, az IPC-t.

Ajánlott irodalom

Andrew S. Tannenbaum – Albert S. Woodhull: Operációs rendszerek (Panem, 1999.)
➲ http://mirrors.kernel.org/LDP/LDP/tlk/tlk.html

Garzó András
(garzoand@interware.hu) Körülbelül három éve foglalkozik Linux- és más Unix-rendszerekkel. Legjobban az operációs rendszerek lelkivilága érdekli, de nyitott egyéniség. Kedvenc étele a palacsinta, és van egy Richard nevű macskája. Minden észrevételt, megjegyzést, levelet szívesen fogad.

Clipper-programok Linux alatt

Emlékszik még valaki a Clipper (XBase) fejlesztői környezetre? Az elmúlt két évtizedben többmillió programsor született ezen a nyelven.

Jelentős azoknak a programozóknak a száma, akik jól ismerik, és még ma is készségszinten tudják használni ezt az eszközt. Napjainkban is számos Clipper-alapú rendszert használunk (a legismertebb talán a Volán Elektronika LIBRA ügyviteli csomagja), arra is találunk azonban példát, hogy az MS Visual FoxPróban új fejlesztések indulnak el. A DBase fájlformátum mindenesetre nagyon elterjedt, így minden rendszer alatt léteznek azok a segédprogramok, amelyekkel kezelni lehet őket. Valószínűleg ezek a felismerések keltették életre azt a projektet, amelyik a clip rendszert megalkotta és jelenleg is fejleszti. A clip csomag nagyon dinamikusan fejlődő GNU-program, amelyet 2001 nyarán fedeztem fel a ➲ http://www.linux.org gyűjteményben kutatva. A program és a fejlesztők honlapja a ➲ http://www.english.itk.ru/clipper/index.html címen érhető el, ahonnan mindig letölthető a legfrissebb változat.
A cikk írásának idején a `clip-prg-0.99-1.tgz` volt a legfrissebb csomag.
Annak idején a Clipper-környezet a Windows 9x/NT és a hálózatos (SQL-kiszolgáló) technológia térhódítása következtében került ki a fejlesztők eszköztárából. Miért? A két fő ok valószínűleg a következő volt:

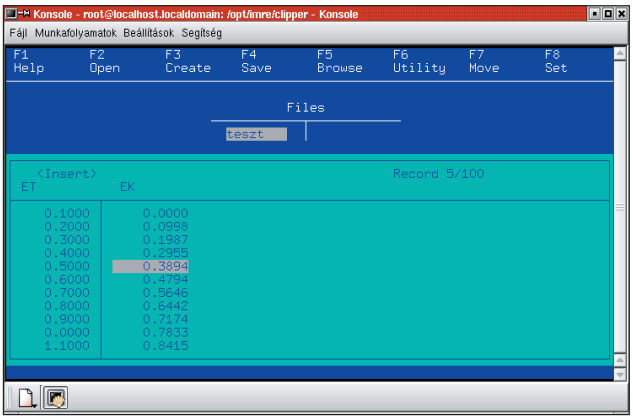
- A Clipper nem volt felkészítve a többrétegű alkalmazások előállítására, ezért az ebben írt programok szükségtelenül terhelték a hálózatot. Az adatok és a Clipper programok általában egy fájlkiszolgálón voltak találhatók, csakhogy a program futtatása és az adatkezelés az ügyfélgépen történt. Ez ebben a formában tarthatatlan volt, hiszen a hálózaton mindig minden adat átment a DBF-fájlok és a program memóriája között.
- A Clipper alapvetően nem grafikus felhasználói felületre lett kitalálva – a puritán kinézetű szöveges képernyők nem feleltek meg a kordivatnak.

A clip rendszer

A clip csomag telepítése után elkezdtem vizsgálni, vajon a létező régi nagyobb Clipper-programjaimat (teljesítményelszámolás, nyugdíjpénztár stb.) le tudom-e fordítani. A másik kérdés az volt, hogy a program használata során a sikeres fordítás és szerkesztés után tapasztalok-e valamilyen hiányosságot. A programok fordítása és szerkesztése a régi `Makefile`-ok átírását jelentette. Számomra hatalmas meglepetés volt, hogy a clip a több tízezer sorból álló programokat kifogástalanul lefordította. A bináris programok kipróbálása során semmilyen hibát nem tapasztaltam, az alkalmazások kifogástalanul működtek. Csodálatos volt arra gondolni, hogy ez a bináris kód valódi, ELF-formátumú Linux-program. A régi rendszerek okozta kellemetlenségek (`CLIPPER`, `FILES` környezeti változók, kevés memória) teljesen eltűntek. Ez a sikerélmény vezetett oda, hogy alaposabban is elkezdtem tanulmányozni a rendszert, amiről kiderült, hogy sokkal többre is képes, mint gondoltam.
A leírás szerint a környezet jelenleg a következő szolgáltatásokat kínálja:

- Az alapcsomag teljesen csereszabatos a CA Clipper 5.3-mal.
- Programozói illesztőfelület létezik az Oracle, PostgreSQL és MySQL adatbázis-kezelőkhöz. Ez lehetővé teszi, hogy adatainkat ne csak *.DBF fájlokban tároljuk, ami meglehetősen izgalmas és új nézőpont az XBase-programozók számára.
- Grafikus felhasználói kezelőfelület- (GUI) csomag (a GTK+ elemkészletre alapozva) használható.
- A TCP/IP csatolója elérhető. Ennek tanulmányozásához a clip SMTP-elem megvalósításának átnézését ajánlom, ami egyébként programjainkból elektronikus levelek könnyű és önműködő küldését teszi lehetővé.
- A `gzip`, `bzip2` tömörített formátumokat támogató csomag.
- A `crypto` titkosított formátumot kezelő csomag.
- Folyamatok kezelése (párhuzamos programozás).
- A soros kapu kapcsolattartását kezelő csomag.
- Teljes nemzeti támogatás (I18N).

A clipet a következő rendszerekre ültették át: Linux, FreeBSD, OpenBSD, Windows 9x/NT/2000.



1. kép A Clipper DBU programjának linuxos változata

A nagyobb programok fordításának és futtatásának bemutatására az *1. kép* szolgál. A kép a Clipper DBU programot futás közben mutatja. A program Clipper nyelven készült és lefordítása semmi gondot nem okozott a clip számára.

Érdekes észrevételek

Régi programjaim clippel fordított változatának nézegetése során rájöttem, hogy a `telnet`, illetve az `ssh` vagy az X-terminál használata során én igazából vékony ügyfél vagyok. Az adatok és a programok emiatt nem terhelik szükségtelenül a hálózatot. Ugye, milyen izgalmas? A Linux egyszerű módon biztosít olyan programhasználati környezetet, ami Windows alatt csak a kétrétegű alkalmazások bevezetésével érhető el. A bináris programok kisméretűek. Ez azért lehetséges, mert nincsenek összeszerkesztve a clip könyvtáraival. A Linux kor-

1. lista Függvénytábla készítése clippel

```
// fuggveny.prg - F ggvØnytÆbla
// lØtrehozÆsa
bKod = { | p1, p2 | 2*p1*p2 }
// Ez egy k dblokk

use TESZT      // a TESZT tÆbla megnyitÆsa
               // rØgi st lusban
zap            // a TESZT tÆbla sorainak
               // t rlØse

x = 0 // Az ET-on fut vØgig
y = 0 // Az EK-en fut vØgig

do while ( x < 10 )
  y = Transzformacio( x )
// alkalmazzuk a f ggvØnyt
  x = x + 0.1
// 0.1 a lØpØsk z
  append blank
//  res rekord lØtrehozÆsa
  replace ET with x        // ki rÆs a
  replace EK with y        // tÆblÆba
end do
use
// a TESZT tÆbla zÆrÆsa
?
return // a program vØge

// Ennek a f ggvØnynek kØsz tj k el
// a tÆblÆjÆt
function Transzformacio( par )
  t := sin( par )
  if ( t < 0 )
     t := 10
     t := Eval( bKod, t, 2*t)
// a k dblokk hasznÆlata (t == 400 lesz)
  end if
return ( t )*
```

szerű felépítésű osztott objektumkönyvtárainak (pl.: ***libclip.so***) használata gazdaságos méretű és gyorsan betölthető programok létrehozását teszi lehetővé.
A kis méretű programok és a Clipper nyelv nem annyira szigorú típuskezelése a jól alkalmazható „ragasztó” nyelvként való használatot is lehetővé teszi. A `run` parancs bármely tetszőleges Linux-parancsot lefuttatja (pl.: `run ls`). Ezt a kellemes tulajdonságot a clip kiváló előfeldolgozója (preprocessor) még teljesebbé teszi. A C/C++ nyelveken megírt kódrészletek egy clipprogramba könnyen beilleszthetők.
A GTK+-ra épülő illesztés szép kinézetű grafikus programok készítését teszi lehetővé, hogy azokat akár távoli X-terminálról (más szóval: X-kiszolgálón keresztül) is könnyedén használni lehessen.
A clip futtatható környezet a Java-bájtkódhoz nagyon közeli elgondolásra épül. A clip forrásprogramok is bájtkódra fordíthatók, amelyeket a clip virtuális gép hajt végre. Ez könnyen megfigyelhető, ha egy clip forrásprogramot C nyelvre fordítunk. Látni fogjuk, hogy a futtató környezet indítása egy új clip virtuális gép létrehozásával indul.

A clip korszerű nyelvi elemei

A Clipper 5.3 már elég fejlett nyelv volt. A régi – DBase-es parancsokat – az előfeldolgozó valódi függvényhívásokká alakította, gondolatvilágában tehát a C nyelvhez áll közel. Ez természetesen azt is megmutatja, hogy a clip milyen fejlett makrózási lehetőségekkel bír.
A `BEGIN SEQUENCE` utasítás korszerű kivételkezelést (Exception handling) valósít meg. Nézzük csak!

```
// KivØtelkezelØs
begin sequence
  ? "Hiba elıtt..."
  hiba = .T.
  if hiba
    ? "HibÆs Ællapot, break..."
    break 3*5 // a break utÆn egy Ætadand
              //  ØrtØk  rhat
  end if
  ? "Ide sohasem j n a vezØrlØs..."
recover using dobott_ertek
  ? "Kezelem a hibÆt...:", dobott_ertek
// a 3*5=15 kiirÆsa
end sequence
```

A fenti példában a `break` (hasonlóan a C++ vagy Java `throw` utasításához) kivételes helyzetet hoz létre, majd eldob egy értéket, mely kifejezés eredménye lehet. A kivétel kiváltása miatt a program a `recover` ágon folytatódik (a C++ vagy Java `catch` ágaihoz hasonlóan). A clip egy beépített `Error` osztályt is tartalmaz, így a `break` a `recover` ág felé akár ezt is dobhatja.
Újdonság a C `switch` utasításának bevezetése, amelyekkel hatékony elágazásokat lehet szervezni, hiszen a program itt a megfelelő ágra való közvetlen ugrással fut. A régi `case` utasítás továbbra is hasznos, néha azonban a `switch` sokkal hatékonyabb. A `case` hátránya, hogy fentről lefelé minden ágra kiszámolja a logikai kifejezés értékét, majd – ha létezik ilyen – az első igaz ágat hajtja végre.
A `@` műveletjel a clipben a referencia szerinti hivatkozást segíti, amiről a C++ és Java-programozók tudják, mennyire hasznos:

```
x = 100
y = @x    // Az y az x egy mÆsodneve lesz
? y       // 100-at fog ki rni
```

A clip új osztályokat is képes bevezetni, amit külön pontban ismertetünk, e helyütt csak annyit említünk meg, hogy a klasszikus `TBrowse`, `TBColumn`, `Get` és `Error` osztályok a Clipper 5.3-ban is léteztek. A nagy hiányosság mindig is az volt, hogy nem tudtunk új típust, azaz osztályt létrehozni. Nos, a clipben ezt is megtehetjük. Sőt, erre épül a rendszerbe vont lehetőségek jelentős része. Jellegzetes módszer a clip fejlesztőitől, hogy valamilyen új szolgáltatáshalmazt egy-egy új osztály bevezetésével tesznek könnyen elérhetővé.

A clip telepítése és fontosabb programjai

A clip jobb megismerése érdekében első lépésként telepítsük Linux-rendszerünkre!
A `clip-prg-<xxx>.tgz` nevű csomagot megtaláljuk a fejlesztők honlapján. A `<xxx>` mindig a pillanatnyi változatszámot jelenti, ami jelenleg `0.99 patchlevel1`. A clip rendszer fordítása és telepítése egyszerű: a telepítés célhelyét a */usr/local* vagy a */opt* könyvtár alatt érdemes kiválasztani. A példában a */opt* helyet választottam.

2. lista A program a Budapest és Esztergom szavakat írja ki a képernyőre

```
////////////////////////////////////////
// OsztÆly Øs objektum
lampa1 := TlampaNew()
// lØtrehozzuk a lampa1 objektumot
lampa2 := TlampaNew()
// lØtrehozzuk a lampa2 objektumot

// tagf ggvØny-h vÆsok, amivel
lampa1:setHely("Budapest")
// beÆll tjuk a lÆmpa helyØt
lampa2:setHely("Esztergom")
// az adattagokhoz k zvetlen l
? lampa1:hely
// nyœlunk hozzÆ
? lampa2:hely
?
return


////////////////////////////
// Egy œj osztÆly
////////////////////////////
function TLampaNew()

  obj := map() // egy œj objektum

// kØt adattagja
  obj:szin := "piros"
// lesz ennek az osztÆlynak
  obj:hely := "ismeretlen"

// a tagf ggvØnyeknek itt csak a neve adott
  obj:setHely  := @setHelyF()
  obj:nextSzin := @nextSzinF()

return obj // A lØtrehoz  mßvelet visszaadja
           // az œj objektumot

////////////////////////////////////////
// Itt vannak a tagf ggvØnyek megval s tÆsai
static function  setHelyF( pHely )
                 ➥::hely := pHely
return NIL

static function nextSzinF( pnextSzin )
                ➥::szin := pnextSzin
return NIL
```

A fenti csomagot másoljuk be a */opt* könyvtárba, majd adjuk ki a következő parancsot:

```
gunzip  clip-prg-0.99-1.tgz
tar xzvf clip-prg-0.99-1.tgz
```

A forrásprogramok telepítését ezzel befejeztük, helyük a */opt/clip-prg-0.99-1* könyvtár lett. Ebben a könyvtárban többféle `make parancsfÆjl` is található: `mkdeb`, `mkrpm`, `mklocal`. Az első két parancsfájl segítségével Debian vagy Red Hat csomagformátumú, terjesztésre is alkalmas változatot hozhatunk létre. Az `mklocal` parancsfájl a clip rendszert lefordítja és telepíti a helyi gépre. Az `mklocal` lefuttatása esetén a használható, bináris formátumú rendszer a */opt/cliproot* könyvtárban lesz. A telepítés utolsó lépéseként a *.bashrc* fájlban a következő változtatásokat tegyük meg:

```
export CLIPROOT=/opt/cliproot
PATH=$PATH:$CLIPROOT/bin
```

Pillantsunk be a *$CLIPROOT/bin* könyvtárba, ahol a fordítóprogram, a forrásnyelvi hibakereső és számos hasznos segédprogram lakozik. A legfontosabb program a `clip`. Egy **.prg* forrásprogramot a `-e -M` kapcsolóval fordíthatunk le. A `clip` a hivatkozott **.o* objekt és **.a* könyvtárak összefűzését is elvégzi (lásd a rendszerhez mellékelt `Makefile`-okat). A `clip -h` parancs az összes megadható kapcsolót kiírja. A `-b` a hibakiírásokat is beszerkeszti a programba. A `-P` csak a forrásprogram előfeldolgozását (makrók feloldását) végzi el. A `clip_cld` egy forrásnyelvi hibakereső. A további lehetőségek tanulmányozásához a `clip` szolgáltatásainak megvalósításait tartalmazó sok-sok **.prg* és **.c* forrásprogram tanulmányozását ajánlom.

Készítsünk függvénytáblázatot!

Kezdjük a munkát a clip rendszerrel! Első feladatként készítsünk egy függvénytáblázatot, amit a *teszt.dbf* fájlban fogunk tárolni (lásd az *1. képet*). Az `ET`=értelmezési tartomány `(numeric 10, 4)`, az `EK`=értékkészlet `(numeric 10, 4)`. A táblázatban a `TranszformÆci` : ET-->EK függvény értékeit a `(0, 10)` tartományban számoljuk ki.
Az *1. listában* látható program által elkészített táblázat egy részletét az 1. képen is láthatjuk. A *fuggveny.prg* forrásfájl fordítását a `clip -e -M fuggveny.prg` paranccsal végeztük el, ami létrehozza a `fuggveny` nevű futtatható programot.

Kifejezéskiértékelés

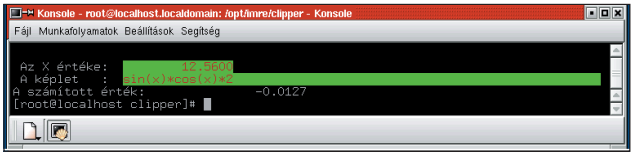
A mai napig kedvelem a Clipper azon tulajdonságát, ami lehetővé teszi, hogy kódrészleteket futás közben fordítsunk le. Ezzel a lehetőséggel könnyen készíthetünk algoritmus-szótárakat is. Az alábbi példa egy kifejezéskiértékelő program, ami az adat- és képletbekérő, valamint az eredménykiíró szakasszal együtt is csak nyolc sor.

```
// makro.prg - Makr helyettes tØs
X = 0.0000
Keplet = "X" + space(50)

@ 1, 1 say "Az X ØrtØke: " get X picture
➥"9999999999.9999"
@ 2, 1 say "A kØplet    : " get Keplet
read
eredmeny = &Keplet
// Itt t rtØnik a kifejezØskiØrtØkelØs
// (az & jel hatÆsÆra)
? "A szÆm tott ØrtØk: ", eredmeny
```

Ugye, milyen egyszerűen oldottuk meg ezt az összetett feladatot? A programot a `clip -e -M makro.prg` paranccsal fordíthatjuk le. A kapott `./makro` program futását a *2. kép* mutatja.
A továbbiakban röviden nézzük meg a clip következő két haladóbb témakörét: az új osztályok készítését, valamint a külső C/C++ eljárások beépítését.

Az objektumközpontú programozás

A clip új lehetősége, hogy teljes értékű új típusokat készíthetünk benne. Példaképpen egy nagyon leegyszerűsített forgalomirányító lámpa osztályának elkészítését mutatom be. Egy új osztály bevezetésének módja engem leginkább a JavaScriptben alkalmazott megoldásra emlékeztet. A *2. listában* látható program egy `TLampaNew` nevű osztályt határoz meg. A `lampa1` és `lampa2` ennek az osztálynak két objektuma. A tagfüggvények, adattagok meghívását ebben a nyelvben az `objektum:tagf ggvØny`, illetve `objektum:adattag`



2. kép Kifejezéskiértékelés

írásmóddal tehetjük meg. A program a Budapest és Esztergom szavakat írja ki a képernyőre.
A kód tanulmányozásával könnyen megérthető egy új osztály kezdeti értékadása és a tagfüggvénynek a megvalósítása.
A `static` minősítő használatával a függvénynevek a fájlra nézve helyiek lesznek, azaz a megvalósítást elrejtjük a külvilág elől. Ezt célszerű így kódolni, hiszen a tagfüggvényeket úgyis csak az objektumokon keresztül fogjuk meghívni. Az eljárásokat megvalósító függvényeket és az objektumbevezetés során megadott tagfüggvényneveket a `@` műveletjel használatával kötöttük össze. Ez azt jelenti, hogy az `obj:setHely` eljárás egy másodnév a `setHelyF()` függvényre. A `":"` érvényességi kör műveletjel 2. listában való használata azt pontosítja, hogy például a `szin` nem helyi változó, hanem osztályunk egyik adattagja.
A forráprogramok ***clip/classes*** és ***cliplibs*** könyvtárainak programjait mindenki kedvére nézze át, mert sokat lehet tanulni belőle. Egyrészt több érdekes cliposztályt ismerhetünk meg behatóbban, másrészt azt a tudást is finomíthatjuk, amivel saját osztályokat készíthetünk.

Külső C/C++ nyelvű függvények beillesztése

A clip egyik erőssége, hogy könnyedén összeilleszthető vele egy C/C++ nyelven megvalósított algoritmus. Ez azért olyan fontos, mert C/C++-ban a teljes Linux-szolgáltatáskör elérhető. A bonyolult eljárásokat például a C++ ***STL*** (Standard Template Library) könyvtár használatával írhatjuk meg. Tekintettel arra, hogy a C/C++ eljárások előnye nem kíván külön magyarázatot, térjünk át inkább arra, hogyan kell ezt a lehetőséget használni. Első feladatunk egyszerű lesz. Írunk egy függvényt, ennek egyetlen `string` értéke lesz, amit átvesz a hívótól, majd vissza is adja neki.

```
// teszt.c
#include <errno.h>
#include <limits.h>
#include <string.h>
#include "clip.h" // Ez egy clippel szÆll tott
                  // fejÆllomÆny

int clip_TESZT_CFGV( ClipMachine *mp )
{
  _clip_retc(mp, (char *)_clip_parc(mp, 1) );
      return 0;
}
```

A `clip_TESZT_CFGV` függvényt a clip programból `TESZT_CFGV` néven hívjuk meg, azaz a „clip_” előtagot elhagyjuk. Vegyük észre, hogy C függvények értékadása mindig olyan, hogy értékként a futtató clip virtuális gépre irányított mutatót adjuk át. A visszatérés típusa mindig `int`. Nézzük meg a fenti függvény clip programbeli használatát is:

```
// TesztKulsoC.prg
extern TESZT_CFGV
// Ebben a f ggvØnyben Ætadunk egy sz veget,
// amit a f ggvØny lefutÆsakor visszakapunk.
? TESZT_CFGV("Ez egy pr basz veg...")
0?
```

A `./TesztKulsoC` program lefutásakor „Ez egy próbaszöveg…” mondat íródik ki a képernyőre.
Egy pillanatra tekintsünk ismét a `clip_TESZT_CFGV` C függvényre. Látható, hogy a értékek átvétele és átadása a `_clip_par?` és `_clip_ret?` függvények (a ? helyén most „c” van, ami a karakteres típusra utal) használatával történik. Ezen függvények első értéke szintén mindig a clip gépre mutató cím. A második érték egy szám, ami azt mondja meg, hogy a függvénynek átadott értékek közül hanydikat akarjuk lekérdezni.
Most megmutatom, hogyan kell lefordítani és összeszerkeszteni ezt a programot.
A C forrásfájl fordítása a ***gcc -c teszt.c*** parancsal történik (eredményül a ***teszt.o*** objektumfájlt kapjuk).
A futtatható program létrehozása a `clip -e -M TesztKulsoC.prg teszt.o` paranccsal történik.
Ennyi ismerkedés után nézzünk példát arra, hogy a clip fejlesztői miként valósították meg a clip belső `sin(x)` függvényét. A következő listán látható részlet a ***clip _math.c*** forrásfájlból származik.

```
// A clipbıl  gy h vjuk: x = sin(3.45)
int clip_SIN(ClipMachine * mp)
{
  int len, dec;
// egy double ØrtØk ÆtvØtele
  double d = _clip_parnd(mp, 1);
// a tulajdonsÆgok lekØrØse
// (hossz Øs tizedesek)
  _clip_parp(mp, 1, &len, &dec);
  dec = mp->decimals;
  _clip_retndp(mp, sin(d), len, dec);
//  rtØkk ldØs a h v nak.
  return 0;
}
```

A C++-ban írt függvények esetén ne felejtsük el, hogy kódoltan vannak jelen az ***object*** fájlban, így azokat az eljárásokat, amelyeket a clipből meg szeretnénk hívni, `extern "C"` jelzéssel lássuk el – ez megakadályozza a nevek kódolását.

Nyíri Imre
(inyiri@mol.hu) jelenleg a MOL Rt.-nél dolgozik. Informatikai vállalkozásában az Internet, a Linux, valamint a Java programozás gyakorlati hasznosításával foglalkozik, örök szerelme a C++ maradt.

A GNU dbm adatbázis-kezelő

Gyakran megesik, hogy nincs szükség SQL-lekérdezésekre, bonyolult adattáblákra, csak néhány kulcs-érték párra és sebességre. Ilyenkor jöhet jól a GDBM.

Amennyiben C-ben programozol, és égető szükséged lenne néhány változóhoz tartozó érték tárolására, első ötleted bizonyára egy egyszerű szöveges állományban történő tárolás. Ennek azonban több hátránya is van. A szöveges állományban való keresés nagyon lassú, főleg ha nem egy-két rekordról, hanem sokkal többről van szó. Másrészt e módszer használatával lemondhatsz a bináris adatok tárolásáról. Ha igazi programozó vagy, akit nem hagynak nyugodni az ehhez hasonló kérdések, biztosan eltöprengsz rajta, hogyan tudnád valamilyen saját formátumban megoldani a dolgot. Teljesen felesleges, ugyanis a GDBM-et neked találták ki!

A GDBM egy függvénykönyvtár, amellyel a fentebb említett kulcs-érték párok tárolásának gondját lehet megoldani. Ha már programoztál Perlben, biztosan hallottál a hashről. Nos, azok az adatbázisfájlok, amelyeket így hozol létre, szintén hashek. A GDBM, hogy visszafelé is csereszabatos legyen, a régebbi ***dbm***, illetve ***ndbm*** könyvtárak függvényeit is tartalmazza. A GDBM fontos újítása, hogy az elődökkel szemben a segítségével tetszőleges hosszúságú kulcs-, illetve értékpárokat lehet tárolni.

Első lépésként telepítened kell a GDBM-et. Ha Debian-felhasználó vagy, a `libgdbmg1`, illetve a `libgdbmg1-dev` csomagokban található változatot használhatod. Én mégis inkább azt javaslom, hogy a forrást töltsd le, és mindig a legfrissebb változatot fordítsd le. Az általam használt Woodyban még egy 1994-es változat szerepelt. Az igazsághoz az is hozzátartozik, hogy semmi gondom nem volt vele, és ugyanúgy működött, mint a legfrissebb. Mindenesetre a legújabbat ajánlom. A projekt honlapját a ➲ http://www.gnu.org/software/gdbm címen érheted el. Nem túl bőbeszédű, de lényegretörő. Végül is keresned kell egy hozzád közel eső GNU ftp-tükröt, és a ***gdbm*** könyvtárból le kell töltened a legújabb forrást (jelenleg az 1.8.0-s). Magyar ftp-helyet nem találtam, ezért egy osztrák kiszolgálót választottam. Ha lejött, a ***/usr/src*** könyvtárban a következő sorok beírásával csomagold ki:

```
cp gdbm-1.8.0.tar.gz /usr/src
cd /usr/src
zcat gdbm-1.8.0.tar.gz | tar xv
cd gdbm-1.8.0
```

Most jöhet a megszokott `./configure`. A Debian-felhasználók többsége már megszokásból odaírja a végére, hogy `--prefix=/usr`. Ha nem így teszel, a programok többsége a ***/usr/local*** alá települ. A GDBM telepítése folyamán szerintem érthetetlen és zavaró apróság volt, hogy hiába adtam meg a `configure`-nek az előtagot, a `Makefile`-ban továbbra is a ***/usr/local***-ban állt. Akit zavarnak az ilyen finomságok, sajnos kénytelen lesz átírni a `Makefile` 34. sorát.

Ezután a szokásos `make`, majd `make install` következik. Ha programod fordításakor valamit elrontottál, a gcc `undefined reference` hibaüzenettel le fog állni. Ebben az esetben kezdd elölről a folyamatot, és lépésről lépésre ellenőrizd, hogy a leírtaknak megfelelően jártál-e el. Ha elvesznél egy-egy parancs kimeneténeken tengerében, és nem tudod, hogy sikeresen futott-e le, add ki az `echo $?` parancsot. Ez a program legutóbbi visszatérési értékét mutatja. Amennyiben nem nulla, valami hiba történt.

1. lista

```
/*** setmailaddr.c ***/
#include <stdio.h>
#include <string.h>
#include <gdbm.h>
#include <sys/stat.h>

#define ADATBAZIS "emails.db"

int main(int argc,char **argv) {
     datum kulcs,ertek;
     GDBM_FILE dbf;
     printf("%s\n",gdbm_version);
     if (argc!=3) {
       printf("HasznÆlat: %s {nev}
       ➥{email}\n",argv[0]);
       return 1;
     }
     if (NULL==(dbf=gdbm_open(ADATBAZIS,512,
       ➥GDBM_WRCREAT,
       S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH,0))) {
       fprintf(stderr,"Nem siker lt
       ➥megnyitni az adatbÆzist.\n");
       return 1;
     }
     kulcs.dptr=argv[1];

kulcs.dsize=(strlen(argv[1])+1)*sizeof(char);
     ertek.dptr=argv[2];

ertek.dsize=(strlen(argv[2])+1)*sizeof(char);
     if
(gdbm_store(dbf,kulcs,ertek,GDBM_REPLACE)) {
       fprintf(stderr,"Nem siker lt
       ➥eltÆrolni az adatot.\n");
       gdbm_close(dbf);
       return 1;
     }
     printf("%s: %s sikeresen tÆrolva.\n",
             kulcs.dptr,ertek.dptr);
     gdbm_close(dbf);
     return 0;
}
```

A legegyszerűbb, ha egy példán keresztül mutatom be a könyvtár használatát. Írjunk egy egyszerű programot, ami a neveket és a hozzájuk tartozó levélcímeket egy adatbázisban tárolja. Ezután írjuk meg a párját is, ami ugyanezeket az adatokat kiolvassa. Lássuk az elsőt a *listát*, és haladjunk sorról sorra!

Elsőként a szokásos fejlécállományok láthatók. A ***gdbm.h***-ra mindenképpen szükség van, ha a GDBM-et akarod használni. A ***sys/stat.h*** az `open()`-nél már megszokott meghatározások használatához nélkülözhetetlen (lásd később).
A GDBM a kulcsot, illetve az értéket egy-egy szerkezetben tárolja. E szerkezet a ***gdbm.h***-ban van meghatározva:

```
typedef struct {
        char *dptr;
        int dsize;
} datum;
```

A `dptr` mutató az adatra, míg a `dsize` az adat mérete bájtban megadva. Így a `datum`-adattípussal egy kulcs, illetve egy érték-változót hozunk létre.
Az adatbázis megnyitásához egy olyan szerkezet mutatójára van szükség, amit a könyvtár függvényei majd különböző belső célokra használhatnak. Ezt szolgálja a `main` függvény második sora.
Ezután kiíratjuk a használt GDBM-könyvtár változatát. A `gdbm_version` állandó tartalmazza ezt a karakterláncot. Ha nincs meg a két érték (név és e-mail), a program futása megszakad.
Ezt követően az adatbázist a `gdbm_open` függvény segítségével megnyitjuk. Az első érték az állomány neve (ebben az esetben `emails.db`). A második a `block_size`: ez az egyszerre beolvasandó bájtok számát határozza meg, ami legalább 512, illetve ennek valamely egész számú többszöröse. A harmadik érték mondja meg, hogy a folyamat milyen szerepet fog kapni az állománnyal való munka során. A folyamat író vagy olvasó lehet. Ha egy folyamat írja az adatbázist, akkor ugyanabban az időben senki sem fordulhat az állományhoz. Ha egy folyamat olvassa, akkor más folyamatok olvashatják vele párhuzamosan, de senki sem írhatja. Mindezek alapján itt a következő meghatározásokat használhatjuk:

`GDBM_READER`	folyamatolvasó;
`GDBM_WRITER`	folyamatíró;
`GDBM_WRCREAT`	folyamatíró; ha nem létezik, létrehozza;
`GDBM_NEWDB`	folyamatíró; új adatbázist hoz létre akkor is, ha már létezik.

A negyedik érték egy szám, amelynek akkor lesz jelentősége, ha új állományt kell létrehozni, ugyanis ennek jogosultságait jellemzi. A meghatározások jelentéseiért olvasd el az `open` súgóoldalát (`man 2 open`). Én a felhasználónak olvasási-írási jogot adtam, a csoportnak és a többieknek csak olvasásit.
Utolsó értéknek egy függvény adható meg, ami hiba esetén lefut. Ha 0-t adsz meg, a könyvtár alapértelmezett függvénye fut le. A `gdbm_open` visszatérési értéke `GDBM_FILE`-típusú, illetve `NULL`, ha nem sikerült megnyitni.
Ezt követően a `kulcs`, illetve `ertek` szerkezeteket a programnak átadott értékeknek megfelelően töltjük fel. A `dptr` egy az egyben a megfelelő érték mutatója, míg a `dsize` a szöveg hossza. Azért adtam hozzá egyet a `strlen()`-hez, mert a karakterláncot lezáró `\0`-át nem tartalmazza, így az nem kerülne tárolásra.
Most következik a program érdemi része, amely a kulcs-, illetve érték-párt tárolja. A `gdbm_store` értékei: a megnyitott adatbázis, a kulcs, illetve az érték (ezek `datum`-típusúak). Az utolsó érték kétféle lehet:

`GDBM_INSERT`	ha a kulcs már létezik, hibával tér vissza;
`GDBM_REPLACE`	ha a kulcs már létezik, a rekordot felülírja.

Két érték nem szerepelhet egyazon kulccsal az adatbázisban, ezért feltétlenül meg kell határozni, hogy a függvény miként viselkedjen. Ha a függvényt egy olvasó folyamat hívta meg, `-1`-gyel fog visszatérni. Ha a `GDBM_INSERT`-tel hívod meg és a kulcs már létezik, akkor `1`-et ad vissza. Ha minden rendben ment, a visszatérési érték `0`.
Végül értesítjük a felhasználót a sikeres tárolásról, és lezárjuk az adatbázist (`gdbm_close`).
A létrejött állomány egy kulcs esetén is 1,5 Kb, ráadásul „szabad szemmel” olvashatatlan. Ugyanakkor nagyon gyors, és rövidesen meglátod, milyen egyszerűen nyerhető ki belőle az adat. Még egy pillanat erejéig érdemes elidőzni azon, mit mond rá a `file` parancs:

```
emails.db: GNU dbm 1.x or ndbm database,
➥little endian
```

Most jöjjön a második program, amely a tárolt neveket és címeket olvassa ki! (2. lista, 36 CD Magazin/GNUdbm)
Most már csak nagy vonalakban tekintsük át, hogy a program mit is csinál.
A `gdbm_open` negyedik értéke jelenleg `0`, mert az adatbázist olvasásra nyitom meg, újat nem hozok létre.
Ezt követően egy `if-else` szerkezettel eldöntöm, hogy a programnak vannak-e értékei. Ha vannak, a megadott kulcsokhoz tartozó értéket íratom ki, ha nincsenek, az összes kulcsot kiíratom.
Az első ágban egyből új függvényeket láthatsz: `gdbm_firstkey` és `gdbm_nextkey`. A GDBM ezekkel a függvényekkel segíti a programozót, hogy az összes kulcson végig tudjon menni.
A kulcsok egymásutánisága ugyanakkor nem biztosított! Semmi sem határozza meg, hogy milyen sorrendben kell következzenek. Az adatbázis megváltozásával a sorrend másként alakulhat. Ha mindkét programot lefordítottad, adj hozzá új neveket, majd az összeset írasd ki. Látni fogod, hogy amit először vettél fel, nem feltétlenül az első lesz a sorban.
A példa végül is önmagáért beszél, még a `gdbm_fetch`, illetve a `gdbm_exists` függvényekről érdemes egy-két szót ejteni.
A `gdbm_fetch` a megadott kulcshoz tartozó értéket adja vissza a `datum` szerkezetben. Ha a kulcs nem létezik, a visszakapott `datum` szerkezet `dptr` mutatója `NULL` értéket fog kapni. Ezért előtte a `gdbm_exists` függvénnyel érdemes ellenőrizni, hogy a kulcs létezik-e. Fontos még tudni, hogy a `gdbm_fetch` által visszaadott `datum` szerkezet `dptr` mutatója által hivatkozott memóriaterületet a könyvtár a `malloc` függvénnyel foglalja, viszont önműködően nem szabadítja fel. Erre neked, a programozónak kell ügyelned.
Most pedig mindkét programot fordítsd le az alábbi minta szerint:

```
gcc -Wall -o akarmi akarmi.c -lgdbm
```

A `-Wall` az összes figyelmeztetést megjeleníti. Elvileg egyet sem szabadna kapnod. A fordítás után a futtatható állományt a `-o` kapcsoló után álló név alatt érheted el. Végül a `-lgdbm` utasítja a gcc-t, hogy a fordítás végeztével a GDBM könyvtárat fűzze össze. Ez feltétlenül szükséges, különben `undefined reference`-t kapsz!
A sok munka után jöhet a megérdemelt játék. Vegyél fel neveket, írasd ki őket, és figyeld meg, hogyan változik az adatbázis mérete, illetve az elemek sorrendje!
Jó szórakozást!

Fülöp Balázs
(xut@freemail.hu) 17 éves, imádja a Túró Rudit, a Debian Linuxot és a teheneket. Az ELTE Radnóti Miklós Gyakorlóiskola tanulója immár ötödik éve. Kedvenc írója Slawomir Mro□ek. Leginkább a számítógépes hálózatok biztonsága érdekli.

A Python közvetítői

Készítsünk a Python és a közvetítőminták segítségével párbeszédablakokat!

Bonyolultabb párbeszédablakok fejlesztésekor a végére mindig teljes lett a zűrzavar, emiatt az elemeket el akartam választani egymástól. Azt szerettem volna elérni, hogy az elemek állapotainak kiértékelése egy központi helyen történjék, így változtatás esetén a kódot csak egyetlen helyen kellene módosítani. Egy csellel sikerült eldöntenem a kérdést. Első lépésben azzal próbálkoztam – és biztos vagyok benne, hogy más is így tett volna –, hogy szétnéztem a Weben, hátha valaki más már készített ehhez hasonlót. Így jutottam el egy tervezési mintákkal foglalkozó vitafórumra, melynek témája elsősorban egy könyv köré szerveződött, ***Design Patterns: Elements of Reusable Object-Oriented Software*** a címe. Ezt a könyvet én is csak ajánlani tudom mindenkinek, akik hatékonyabban szeretnének programozni. Bár a könyv a C++-on alapul, a leírtak minden objektumközpontú programozási nyelvre érvényesek, beleértve a Pythont is. A mi gondunkra a ***Közvetítőminta*** (vagyis a Mediator pattern) jelenti a megoldást. Ez a minta teszi lehetővé az elemek csoportjainak központosított irányítását.

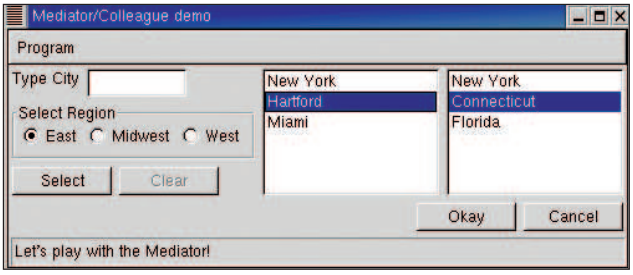
A dolgokat objektumközpontú megközelítésben szemlélve: létezik egy Közvetítőobjektum (mediator), amely mindazokat az objektumokat tartalmazza, amelyeket együttműködésre szeretnénk késztetni; ezeket „munkatársaknak” hívjuk. A Munkatársak egy közvetett hivatkozást tartalmaznak a Közvetítőobjektumra, egymásról viszont nem tudnak. A Közvetítőobjektum erős kötésekkel kapcsolódik minden egyes Munkatársobjektumhoz (collegue), és közvetlenül képes módosítani őket, így érve el, hogy a kívánalmaknak megfelelően viselkedjenek. Nekünk pedig pontosan erre volt szükségünk; a közvetítő központosítja az objektumok kezelését és csökkenti az objektumok közti kötések számát.

A ***Közvetítő/Munkatárs*** mintához azonos felületen keresztül férhetünk hozzá, melyen keresztül valódi osztályobjektumokat képezhetünk. A Közvetítőfelületnek van egy `ColleagueChanged()` tagfüggvénye, ezt – ha megváltoztak – a munkatársak hívják meg. A `MunkatÆrs` (`Colleague`) felületnek mindössze egyetlen szükséges tagfüggvénye van, a `Changed()`, melyet a származtatott objektumok hívnak meg, így értesítve a közvetítőt a változásról. Ezen kívül a Munkatársosztály egy `mediator` nevű nyilvános adattaggal is rendelkezik, mely a Munkatársat tartalmazó Közvetítőosztályra tartalmaz hivatkozást.

Mindez nagyon szép, de hogyan hozzunk létre olyan párbeszédablakokat, amelyek ténylegesen megvalósítják ezt a ***Közvetítő/Munkatárs*** mintát? A Python objektumközpontú lehetőségeit fogjuk kihasználni, a grafikus felület kezeléséhez pedig `wxPython`-t alkalmazunk. Elsőként hozzunk létre egy Közvetítőosztályt:

```
class Mediator:
    def __init__(self):
        pass

    def ColleagueChanged(self, control, event):
        self._ColleagueChanged(control, event)

    def _ColleagueChanged(self, control, event):
        pass
```

Ebben a példában a Közvetítőosztályt Sablon (Template) mintaként hoztuk létre, így választva el az osztály felületét az osztály tagfüggvényeinek megvalósítási részétől. Ennek az osztálynak a felhasználói a `CreateColleagues()` tagfüggvényt hívják meg, a származtatott osztályokban azonban a `_CreateColleagues()` tagfüggvényt írjuk felül. A sablonmintát a továbbiakban már nem tárgyaljuk, de megemlítem, mert ha nem sajnálunk egy kicsit utánajárni, a sablonoknak is nagy hasznát vehetjük.



Közvetítőobjektum használat közben

Most pedig hozzuk létre `Collegue` osztályunkat:

```
class Colleague:
    def __init__(self, mediator):
        self.mediator = mediator
    def Changed(self, colleague, event):
      self._Changed(colleague, event)

    def _Changed(self, colleague, event):
      self.mediator.ColleagueChanged
➥(colleague, event)
```

A `Collegue` osztály is egy sablonmintán alapul. Miként az előbbiekben, a felhasználóknak ebben az esetben is a `Changed()` tagfüggvényt kell meghívniuk, viszont a `_Changed()` tagfüggvényt kell felülírniuk, ha szükséges. Ezen kívül a `MunkatÆrs` osztály is rendelkezik egy adattaggal (`self.mediator`), amely a közvetítő objektum példányára hivatkozást tartalmaz hivatkozás. Ez továbbadódik a munkatársobjektum létrehozójának.

Mivel példaprogramunkkal csak a ***Közvetítő/Munkatárs mintát*** szemléltetjük, némileg erőltetettnek hathat. Az egyszerűség kedvéért a példában a `wxDialog` helyett a ***wxPython*** könyvtárban található `wxFrame`-et használtam fel. Egyébiránt a kód ugyanaz. Mivel példánkban a `wxFrame` tartalmazza elemeinket, ennek jut a közvetítő szerepe is. Ahhoz, hogy a Közvetítőosztály felületét a `wxFrame`-ből fel tudjuk használni, `MainFrame` néven létre kell hoznunk egy új osztályt, mely a következőképpen fest:

```
class MainFrame(wxFrame, Mediator):
 def __init__(self, parent, ID, title):
   wxFrame.__init__(self, parent, ID, title,
```

```
    wxDefaultPosition, wxSize(400, 300))
    Mediator.__init__(self)
```

Ebben a kódban létrehoztunk egy új osztályt, mely a `wxFrame` és `Mediator`, vagyis `K zvet t1` osztályoktól egyaránt örököl. Hogy ez a Pythonban megfelelően működjön, `MainFrame` osztályunk `__init__()` tagfüggvényéből mindkét szülőosztály létrehozóit külön meg kell hívnunk. Ahhoz, hogy a `MainFrame` osztály a benne rejlő elemekkel mint munkatársakkal tudjon kapcsolatot tartani, az egyes elemeket a Munkatársosztálytól kell származtatnunk. Példaként hozzunk létre egy szöveges elemet, mely `MunkatÆrs` objektum is egyben. Ehhez egy új `myTextCtrl` osztályt kell létrehoznunk:

```
class myTextCtrl(wxTextCtrl, Colleague):

 def __init__(self, mediator, *_args, **_kwargs):

     apply(wxTextCtrl.__init__,
          (self,) + _args, _kwargs)
      Colleague.__init__(self, mediator)
```

Ily módon létrehoztunk egy új osztályt, mely egyaránt örökli a `wxTextCtrl` és `Colleague`, vagyis a Munkatársosztályok tulajdonságait. Még egyszer: hogy ez megfelelően működjön, mindkét szülőosztály létrehozóját külön-külön meg kell hívnunk. Mint az előbbiekben, ezt most is az `__init__()` tagfüggvényből tesszük meg. Az `apply()` függvénnyel biztosítjuk, hogy adjon át a `wxTextCtrl` létrehozójának minden értéket megfelelően. A Munkatársosztály `__init__()` tagfüggvényét közvetlenül hívjuk meg, átadva neki a *`mediator`* hivatkozást. Így most létezik egy osztályunk, mely `Collegue` és `wxTextCtrl` is egyben. Ennek az osztálynak a példányai képesek fogadni a `wxTextCtrl` eseményeit, ugyanakkor a `Collegue` osztály jellemzőivel is rendelkeznek. Amennyiben egy `MainFrame`-et érintő esemény keletkezik, az eseménykezelő meghívja a `Changed()` tagfüggvényt, és továbbítja az önmagára mutató hivatkozást, valamint az eseménnyel kapcsolatos értékeket. Amint azt a `Collegue` osztályban megadtuk, a `Changed()` tagfüggvény meghívja az elem közvetítőjének `ColleagueChanged()` tagfüggvényét. Ily módon a `MainFrame` objektum (mely egyben `K zvet t1` objektum) értesül az elemen bekövetkező változásokról.

Hogyan kapcsoljuk össze mindezt a ***MainFrame*** ablakban? Elsőként – mint ahogyan a `myTextCtrl`-lal tettük – az összes felhasználandó elemből létre kell hoznunk egy származtatott osztályt, amelyeket egyúttal a `Collegue` osztályból is származtatunk. Ezt követően pedig, mint a legtöbb ***wxPython*** ablak esetén, az elemeket ablakunk létrehozójában életre hívjuk – esetünkben a `MainFrame __init__()` tagfüggvényében. Minden egyes alkalommal, ha létrehozunk egy elemet, a `MainFrame` osztály hozzáadja magát a származtatott elem értéklistájához. A 36. CD Magazin/Python könyvtárában található egy példaprogram, amely ezt a dolgot sokkal részletesebben tartalmazza. A `MainFrame.__init__()` tagfüggvényben ezúttal jóval több dolgot láthatsz, de ne csüggedj, mivel a kód nagy része a `wxPython` működéséért felelős, és használata nem feltétlenül kötelező, csak a felületet teszi barátságosabbá.

A `MainFrame.__init__()` tagfüggvényben létrehoztam egy `self.__colleagueMap` nevű `dictionary`-t, mely az ablakban található `Collegue`-elemekre és azok tagfüggvényeire tartalmaz hivatkozásokat, kulcs/érték párokba rendezve. A Python nem rendelkezik olyan `switch/case` utasítással, mint a C/C++, viszont az éppen szükséges tagfüggvényt a `dictionary`-n keresztül elegánsan meghívhatjuk, ha valamelyik `Collegue` objektumon változás áll be, anélkül, hogy egy hosszú `if-else` szerkezettel kellene bajlódnunk. Példaprogramunk `_ColleagueChanged()` tagfüggvényében láthatsz erre egy példát:

```
def _ColleagueChanged(self, colleague, event):
 if self.__inProcess != true:
     self.__inProcess = true

         if self.__colleagueMap.has_key
         ➥(colleague):
           self.__colleagueMap[colleague]
           (event)
           self.__inProcess = false
```

Ebben a kódban a `colleague` érték arra szolgál, hogy megnézzünk, a nemrég létrehozott `dictionary` tartalmazza-e az adott `Collegue`-at, és amennyiben igen, meghívja a megfelelő tagfüggvényt, és az eseményt értékként átadja. Ily módon tehát nagyon könnyedén hozhatunk létre többirányú elágazást, csakúgy, mint a `switch-case` esetében tettük volna. `Mediator` és `Collegue` objektumaink most már léteznek, és egymáshoz vannak kapcsolva. A `K zvet t1` objektum (`MainFrame`) ezáltal minden a `Collegue` objektumokkal (az elemek) kapcsolatos változásról értesül. Mi maradt még hátra? Létre kell hoznunk a központosított kódot, amellyel az elemek közötti együttműködést biztosítjuk. Ezt a `self.__colleagueMap dictionary`-ben elhelyezett tagfüggvényekben tehetjük meg. Minden egyes tagfüggvényben elhelyezzük a szükséges kódot, amely az egyes eseményekre válaszolt fog adni. Mivel ezek a tagfüggvények `Mediator` objektumunk (`MainFrame`) részei, az ablakban lévő összes elemről tudnak, és módosítani is képesek őket.

A példaprogramot a Python 2.1-es és 2.2-es változataiból próbáltam ki – a hozzájuk tartozó `wxPython`-nal. Ha elindítod a programot, képernyődön a *képen* láthatóhoz hasonló ablak fog megjelenni.

Az összetevők közti együttműködésben az ablakban látható legtöbb elem résztvesz. Ha beírsz egy karaktert a szövegmezőbe, a program gyorskeresést végez, és a listában kijelöli az első illeszkedő elemet. Ezenkívül a ***Kijelöl*** és ***Töröl*** gombok is engedélyezve lesznek. Ha kijelölsz egy várost vagy államot, egy másik ablakban ennek megfelelően rendeződnek át az elemek. Ha a kapcsológombokkal egy másik régiót választasz, a listamezők tartalma visszaáll a kiindulási állapotra, a ***Kijelöl*** és ***Töröl*** gombok pedig újból letiltódnak. Ha a ***Töröl*** gombra kattintunk, törli a szövegmező tartalmát, ugyanakkor letiltja önmagát. Az elemek együttműködéséért felelős kód tehát a `MainFrame` osztályban van, és az elemek nincsenek egymáshoz kapcsolva.

A ***wxFrame*** ablakunk nem tesz semmi hasznosat, viszont kitűnően szemlélteti, hogyan vezényelheti egy `Mediator`-objektum az ablakban található elemek működését. Ennek majd akkor veszed igazán hasznát, amikor egy sokkal bonyolultabb párbeszédablakot kell megtervezned.

Linux Journal 2002. június, 98. szám

Doug Farrel
a Scholastic Inc-nél dolgozik mint vezető programozó. Címhivatkozásokra épülő webes alkalmazások fejlesztésével foglalkozik. Szabadidejében Susan nevű feleségével kerékpáron gyűri a mérföldeket.

SpamAssassin – segítőtárs a mindennapokban

Könnyű és légáteresztő, a levélszemetet azonban csaknem százszázalékosan nyakon csípi.

A Linuxvilág olvasói táborának igen nagy százaléka valószínűleg azzal kezdi hétköznapjait, hogy otthon vagy a munkahelyén elektronikus postafiókját kezdi böngészni. Az elmúlt időszak kéretlen levélözöne miatt naponta akár tucatszám kapjuk a kéretlen levélszemetet. Elég egy óvatlan mozdulat és elektronikus levélcímünk máris közkincsé válhat. Vajon hogyan is történhet mindez? Például egy weboldal karbantartásával foglalkozunk, és nyilvánosságra hozott címünket egy robot leszedi. Esetleg egy nyilvános levelezési listára postáztunk egy levelet, amelynek irattárából az előbb említett robotprogram csemegézik. Előfordulhat az is, hogy egy feltört szolgáltató címlistájából jutnak hozzá stb. A sor szinte a végtelenségig folytatható.

Miként lehetséges azonban az, hogy a csupán egy-két napja létező címre özönlenek a kéretlen levelek? A kérdés inkább az, hogy miért is káros ez számunkra? Nem is olyan régen azt taglaltam, hogyan oldjuk meg, hogy leveleinkről egy telefon segítségével SMS-értesítést kapjunk. Tehát minél több a kéretlen levél, annál több a fizetett SMS. Ennél sokkal fontosabb szempont, hogy minden egyes levélszemét elolvasása után akár többezer idegsejtünk is odavész, ugyanis a folyamatos idegfeszültség miatt e kincset érő sejtjeink nagy mennyiségben pusztulnak. Az ember és a számítógép kapcsolata a nagyszámú hiba miatt amúgy sem felhőtlen, ezért ezt az amúgy sem rózsás helyzetet nem célszerű tovább rontani. Mennyivel egyszerűbb feltelepíteni egy levélszemétszűrőt, amely megteremtheti számukra a csaknem felhőtlen levelezést!

A SpamAssassin egy olyan szűrőprogram, amelynek beállítását egy teljesen kezdő is bármikor meg tudja tenni. Ugyanakkor finomhangolása megfelelő, sőt még a kéretlennek ítélt levelek sem vesznek el, hanem egy külön erre a célra fenntartott állományba kerülnek, hiszen ki tudja, lehet, hogy egy levélszemétnek nyilvánított levél mégis értéket képviselhet. A legegyszerűbb, ha letöltjük a ➲ http://www.spamassassin.org címről.

A Debian Woody-felhasználók az `apt-get install spamassassin` parancs kiadásával is telepíthetik. Azok, akik forrásból telepítenék: egy rendkívül részletes ***README*** állomány áll a rendelkezésükre, amely lépésről lépésre leírja, hogyan telepítsük. Ezek után nincs másra szükségünk, mint hogy saját könyvtárunkban egy ***.procmailrc***-t helyezzünk el, ugyanis a szűrőprogramot a Procmail fogja meghívni, az összes fogadott levelünk így kerül elemzésre. A következő sorokat tehát a */home/felhasználónevem/.procmailrc* állományban helyezzük el:

```
:0fw
| spamassassin -P

:0:
* ^X-Spam-Status: Yes
caughtspam
```

Így a bejövő levél először átadódik a Spamassassinnek, amely eldönti róla, hogy kiszűrendő levél-e vagy sem. Ezek után a levél fejlécrészébe befűzi a kiértékelés eredményét. Ha kéretlen, a levél tárgya kiegészül a

```
*********SPAM********
```

szöveggel, valamint a levéltestbe egy indoklás kerül, hogy miért is lett kiszűrve, valamint a `X-Spam: Yes` sor is megjelenik benne. Amikor a Procmail a levelet visszakapja, a következő szabályon (`[X-Spam-Status]`) fennakadva a ***Caughtspam*** postaládába kerül. Így nem fordulhat elő, hogy egy ismerősünk fontos levele végképpen kárba vész.

Álljon itt egy jellegzetes levélszemét kiértékelése (lásd a *listán*). Jól látható, hogy a program leveleinket az előre beállított szabályok szerint pontozza, és ha a levélben hivatkozás található vagy az `unsubscribe` felirat szerepel – ami a levélszemétre

```
SPAM: ---------------- Start SpamAssassin results ----------------------
SPAM: This mail is probably spam.  The original message has been altered
SPAM: so you can recognise or block similar unwanted mail in future.
SPAM: See http://spamassassin.org/tag/ for more details.
SPAM:
SPAM: Content analysis details:   (19.4 hits, 5 required)
SPAM: Hit! (2.3 points)  BODY: Gives a lame excuse about why you were sent this SPAM
SPAM: Hit! (0.7 points)  BODY: Talks about email marketing
SPAM: Hit! (1.5 points)  BODY: Asks you to click below
SPAM: Hit! (3.5 points)  URI: URL of page called "unsubscribe"
SPAM: Hit! (1.3 points)  URI: Includes a 'remove' email address
SPAM: Hit! (4.8 points)  BODY: Frontpage used to create the message
SPAM: Hit! (2.1 points)  BODY: FONT Size +2 and up or 3 and up
SPAM: Hit! (0.0 points)  BODY: Includes a URL link to send an email
SPAM: Hit! (3.2 points)  HTML-only mail, with no text version
SPAM:
SPAM: --------------- End of SpamAssassin results ---------------------
```


```
1:Kilép  -:Előző  <Space>:Következő  v:Melléklet  d:Töröl  r:Válasz  j:Köv. ?:Súgó
*2002/04/06 19:28:30 hughescr Exp $)
X-Spam-Prev-Content-Type: text/html
X-Spam-Prev-Content-Transfer-Encoding: 8bit

SPAM: -------------------- Start SpamAssassin results ----------------------
SPAM: This mail is probably spam.  The original message has been altered
SPAM: so you can recognise or block similar unwanted mail in future.
SPAM: See http://spamassassin.org/tag/ for more details.
SPAM:
SPAM: Content analysis details:   (18.7 hits, 5 required)
SPAM: Hit! (2.4 points)  'Message-Id' was added by a relay (2)
SPAM: Hit! (3.8 points)  BODY: Gives instructions for removal from list
SPAM: Hit! (2.7 points)  BODY: Claims you can be removed from the list
SPAM: Hit! (1.5 points)  BODY: Asks you to click below
SPAM: Hit! (1.4 points)  BODY: Claims you can be removed from the list
SPAM: Hit! (-1.6 points) BODY: Contains a claim of copyright
SPAM: Hit! (3.5 points)  URI: URL of page called "unsubscribe"
SPAM: Hit! (1.8 points)  BODY: Tells you to click on a URL
SPAM: Hit! (3.2 points)  HTML-only mail, with no text version
SPAM:
SPAM: -------------------- End of SpamAssassin results ---------------------
-0  --  4/6: Free 4 You              *****SPAM***** Your Freebies Have Arri --  (55%)
```

jellemző –, akkor egy bizonyos pontértéket kap. Mivel nem lenne jó, ha például egy hivatkozás miatt a levelünk levélszemét-értékelést kapna, hiszen ismerőseinktől naponta több tucat értékes hivatkozás futhat be, a kéretlen levél csak akkor lesz bélyegezve, ha több pontban is bűnösnek találtatott. Ilyen pontrendszert mi is bármikor felállíthatunk. Fontos azonban, hogy amennyiben levelezési listákra is feliratkoztunk, ezt vagy a szűrési szabályzatba vegyük bele, vagy ennél jóval egyszerűbb, ha a Spamassassint a Procmail legutolsó soraiba rakjuk, hiszen ilyenkor feltehetően a levelezési listákat már különböző szabályok alkalmazásával leválogattuk, és csak ez után kerül sor a levélszemétszűrésre. Például így:

```
:0
* ^Reply-To: linux@mlf.linux.rulez.org
$MAILDIR/linux/

:0
* ^(From|Cc|To|Reply\-To|Delivered-\To):.
➥*bugtraq@lists.securityfocus.com
$MAILDIR/bugtraq/
```

Ekkor a `mlf` és `bugtraq` listákat leválogatjuk, a levélszemétszűrő sorokat csak utána írjuk.
Ezek után garantáltan nem fogunk idegeskedni. A ***caughtspam*** állományba került leveleket naponta érdemes átnézni, hogy nem akadt-e bele mégis egy rendes levél, a személyes tapasztalatom azonban az, hogy az ember jókat mulat egy-egy elfogott levélen. Jó vadászatot!

Varga S. Csaba
(guska@guska.hu) az 1.1-es Slackware óta linuxozik. Kedvtelései közé tartozik a fotózás és Linux telepítése PDA-kra. Legszívesebben a Gerecsében túrázik a barátaival.

Biztonsági indítólemez készítése

Amikor operációs rendszert telepítünk, és a telepítő megkérdi, hogy akarunk-e biztonsági indítólemezt létrehozni, a helyes válasz: Yes (Igen). Ha valamilyen különös oknál fogva ezt a lépést átugrottuk volna a telepítés során, soha nem késő, hogy visszatérjünk és létrehozzunk egyet.
A lemez elkészítésének egyik oka, hogy amikor a rendszert telepítettük, lehetőségünk volt a LILO-t választani rendszerindító eszköznek. Megszokott körülmények között ez a legkézenfekvőbb. A LILO lehetővé teszi a több rendszermag közül való választást, és a választható rendszerindítást, amellyel a Linux és Windows felváltva történő használata egyszerűen megoldható. Amennyiben a jövőben valamilyen okból a LILO mellőzése mellett döntünk, az indítólemez lehet az eszköz, amellyel a Linux-rendszer indítható marad. Így vagy úgy, de mindenképpen jó, ha a biztonság kedvéért rendelkezünk indítólemezzel.
A következőkben bemutatom, hogyan lehet utólag létrehozni egyet. Szükség lesz egy üres, formázott, nem írásvédett lemezre. Mivel a művelet minden adatot felülír a lemezen, az üres lemezre vonatkozó javaslatom betartásával elkerülhetjük, hogy esetleg fontos adatainkat veszítsük el. Helyezzük a lemezt a meghajtóba, majd adjuk ki a következő parancsot:

```
mkbootdisk magvÆltozat
```

A rendszermagváltozat részt azzal a változattal kell helyettesíteni, amelyikről a rendszert indítottuk. Ezt a parancssorba gépelt `uname -r` parancs segítségével tudhatjuk meg. Amikor próbarendszeremen ezt a parancsot kiadtam, a következő eredményt kaptam:

```
2.2.14-test
```

A `-r` kapcsoló utasítja az `uname` parancsot az operációs rendszer változatszámának közlésére. Ezzel az adattal és a parancs futtatásával létrehozhatjuk az indítólemezt:

```
mkbootdisk 2.2.14-test
```

Megjegyzendő, hogy az `mkbootdisk` parancs egy *vmlinuz-változatszám* nevű rendszermagot, valamint a */lib/modules* elérési útvonalon egy *változatszám* nevű modulkönyvtárat keres. Az előző példa alapján tehát egy */boot/vmlinuz-2.2.14-test* nevű rendszermaggal és egy */lib/modules/2.2.14-test* modulkönyvtárral kellett volna rendelkeznem. Azért említem mindezt, mert a rendszermagot tetszőleges néven menthetjük, azonban a */lib/modules* bejegyzés ettől különbözhet. Jó ötlet *vmlinuz-változatszám* formában menteni az új rendszermagokat a */boot* könyvtár alá.
Tudnunk kell azonban, hogy az `mkbootdisk` parancs nem található meg minden rendszeren. Például Debian próbarendszeremen az `mkboot` parancsot használtam indítólemez létrehozására (bár a rendszer alkalmat adott rá, hogy már a telepítéskor megtegyem):

```
# mkboot
```

Itt egy kisebb nehézség adódott: az `mkboot` parancs a *vmlinuz* állományt (a Linux rendszermagját) a */boot* könyvtárban keresi.

Részlet Marcel Gagné: Linux-rendszerfelügyelet című könyvéből

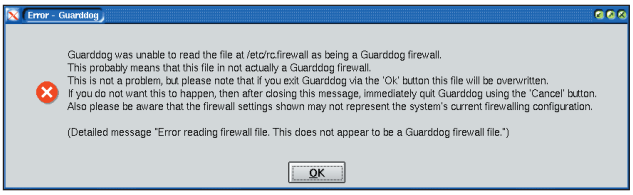
Személyes tűzfalak Linuxon

A mókásan csak aszbesztkabátnak hívott programok GNU/Linux alatt is elérhetőek.

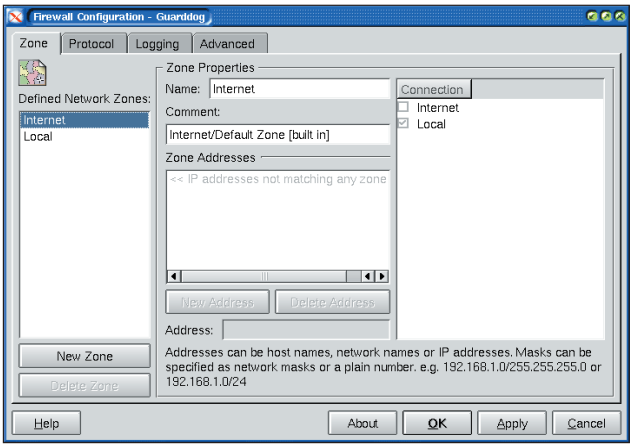
Gyakran, ha a biztonságról esik szó, hiába bizonygatja az ember az átlagosnál többet tudó felhasználóknak, hogy a Linux biztonságosabb, jobb és szebb, mint a jégbe fagyott ablak. Azonnal azt kérdezik: hol van a ***ZoneAlarm*** Linuxra? Sokak számára ugyanis ez a program jelenti a sérthetetlenséget. A kutatások eredményeképpen megismerhettük a placebohatást: pszichikai eredetű betegségekben szenvedőkkel elhitetik, hogy orvosságot kapnak, és ennek következtében hirtelen „meggyógyulnak", ha viszont azt mondják nekik, hogy nem tudnak segíteni, az állapotuk rosszabbodik – csakhogy a bitek világában sajnos nem ér semmit, ha egyre azt mondogatjuk magunkban: nem történt semmi, nem törték fel a gépemet. Akkor is feltörhették, ha azt sulykoljuk, hogy nem történt semmi baj. Szóval nehéz elmagyarázni, hogy a biztonság nem pusztán abból áll, hogy felteszek valamit, ami a kívülről jövő kapcsolatokat engedélyezi vagy éppen letiltja. A ZoneAlarm program viszont nagyon látványos, elindulásakor kis tülkölés hangzik fel, szép kis kijelzőt láthatunk a hálózati forgalomról, mint amilyen például az IceWM tálcáján is található. Egy szó, mint száz, valami hasonlót kell adni felhasználóknak, máskülönben nem békélnek meg a gondolattal, hogy „csak" egy proxy (például Squid) és állapottartó csomagszűrő védi őket a hálózati átjárón az Internet felé. Csupán azon magyarázatunkra csillan fel a szemük, hogy a ZoneAlarm sem több, mint egy csomagszűrő – ezt csak a letölthető változatról állíthatom, ugyanis ezt ismerem –, és ezt akármelyik terjesztés nyújtani képes. Ekkor felvillan a lehetősége, hogy többek között a „gonosz" munkatársak és a rendszergazda ellen beállíthatják maguknak a védelmet, viszont amikor meglátják, hogy mindent kézzel kell megtenni, a Linuxot újra a kőkorszakba sorolják. Erre kellene egy jó megoldás, és ezt a Guarddog program jelentheti. Ebben minden megvan, ami szükséges. Hangzatos név, grafikus beállítófelület a leggyakoribb alkalmazásokhoz. A program a ➲ http://www.simonzone.com/software/guarddog/ címen érhető el.

Telepítés után a programot a `guarddog` paranccsal indíthatjuk a KDE alatti ***K*** menüből, vagy ha más ablakkezelőt használunk, akkor egy grafikus felület alatti konzol alól, mint amilyen az Xterm vagy Eterm programok. Figyelem! Ha nem rendszergazdaként indítjuk, a program tájékoztat minket, hogy beállíthatjuk, milyen védelmet szeretnénk, de nem fog életbe lépni, mert nincs rendszergazdai jogosultságunk. Ezt szemlélteti az *1. kép*. Erre kínál megoldást, ha például az Xterm alatt `su -m` parancscsal (a `-m` kapcsoló hatására grafikus felületet használó programokat is megfelelően tudjuk használni), hogy ekkor mi

1. kép Lényegre törő figyelmeztetés



2. kép Az első indítás öröme



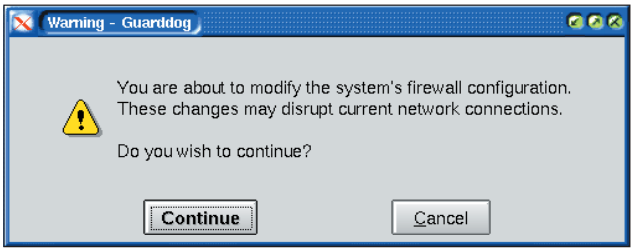
3. kép Kezdődhet a beállítás

történik, a *2. képen* láthatjuk. A képen már rendszergazdai jogosultságokkal rendelkezünk, de mint kitűnik, előtte meg kell adni a rendszergazda (root) jelszavát.

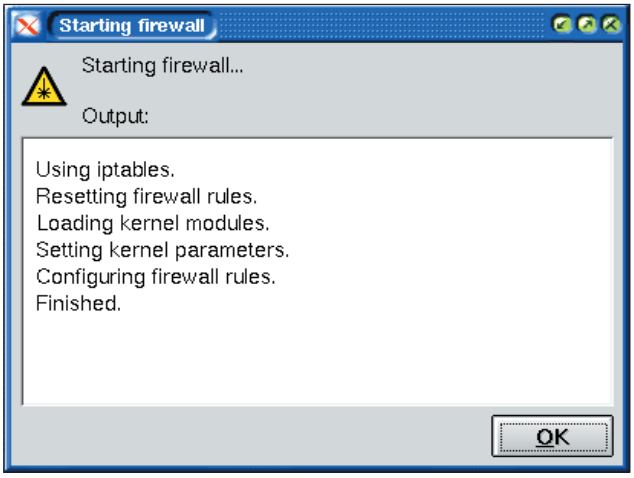
Ha `guarddog`-ot innen indítjuk el, az első alkalommal tájékoztat minket, hogy csomagszűrő tűzfalunk nincs beállítva. Ez természetes is, hiszen most indítjuk először a programot. Nyugodtan haladjunk tovább az ***OK*** gombbal. A *3. képen* látható képernyő fogad minket.

Nézzük végig a füleket!

- ***Zone***
  Zónákat vehetünk fel, illetve törölhetünk, kivéve a két beépítettet: az Internet és a Local nevűt.
  Új zóna felvételénél adjuk meg a nevét (***Name***), valamint a címtartományt, amelyre vonatkozik (***Zone Addresses***). Egész címtartományokat is megadhatunk (`192.168.0.0/24`), de akár egyedi címeket vagy csak egy címet is megadhatunk.
- ***Protocol***
  Ezen a fülön állíthatjuk be a szabályozást. A lenyíló listából válasszuk ki az ***Internet*** zónát. Ezután, mint a 3. képen is láthatjuk, elég egy grafikus menüben kijelölni a használni kívánt alkalmazásokat – valójában a protokollokat, hiszen például az ICQ-protokollt is több program ismeri. Ha egy alkalmazás neve mellett a jelölőnégyzet üres vagy egy x jelet látunk, akkor az le van tiltva, pipa estén engedélyezett. Válasszuk ki, mit szeretnénk használni és engedélyezzük.
  A beállítást azonnal érvénybe léptethetjük, ha az ***Apply*** gombot lenyomjuk. Ekkor a rendszer figyelmeztet, hogy



4. kép Reméljük, nem felejtettünk ki semmit!



5. kép Most már védve vagyunk

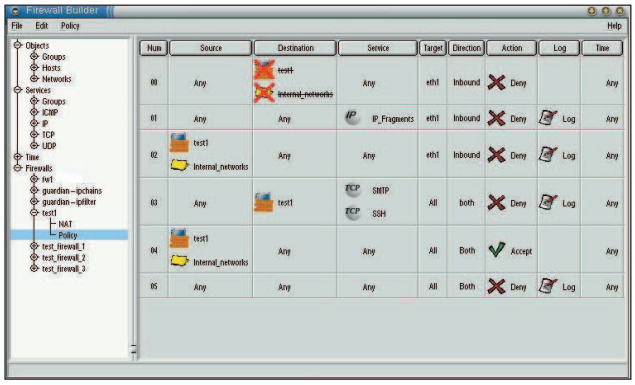
a szabályok életbe léptetésével bizonyos internetkapcsolataink megszakadhatnak. Ez csak akkor fordulhat elő, ha pillanatnyilag olyan alkalmazást is használunk, amelyet az új beállításban letiltunk. Például már nem akarunk többet ICQ-ra csatlakozni, de a program még fut, ekkor a szabályok érvényesítése után kapcsolata megszakad. A figyelmeztetést a következő, *4. képen* láthatjuk.

Az érvényesítés eredményét az *5. képen* látható ablakban tekinthetjük meg. Az ablakban az ***OK*** gombra kattintsunk.

Ekkor a beállításokat a rendszer a kapcsolat típusától függően

- ADSL-vonal estén a ***/etc/ppp/ip-up.local*** fájlba menti, és beszúr egy bejegyzést a parancsfájlba, ami a külső hálózatra csatlakozáskor hívódik meg;
- Más típusú kapcsolat esetén a ***/etc/ppp/ip-up*** fájlba menti, így az előbbi fájlt hívja meg, és a védelem kiépül. Mivel ezek a fájlok minden Internetre történő kapcsolódáskor végrehajtódnak, soha nem fogjuk elfelejteni bekapcsolni, és nem maradunk védtelenek. Érdemes lehet a már elkészített konfigurációt egy fájlba exportálni és későbbi használatra menteni.

- ***Logging***
  Ezen az oldalon a csomagszűrő naplózását állíthatjuk be, valamint a 2.4-es rendszemag lehetőségeit kihasználva korlátozhatjuk, hány csomagot szeretnénk fogadni.
- ***Advanced***
  Itt többek között az ***Export*** gomb segítségével a beállításokat szövegfájlba tudjuk menteni, amelynek mi adhatunk nevet, illetve az ezzel mentett beállítást az ***Import*** gombbal vissza is tudjuk olvasni. Ez a lehetőség akkor hasznos, ha a beállításokat menteni szeretnénk. Ezenkívül lehetőségünk nyílik rá, hogy a tűzfalat letiltsuk (***Disable firewall***), és az eredeti beállításokat visszaállítsuk (***Restore factory defaults***). Fontos még, hogy ha a címet DHCP-n keresztül önműködően kapjuk, be kell kapcsolnunk az ***Enable DHCP on interface*** négyzetet, ahol meg kell adnunk, hogy melyik az a hálózati csatoló, amelyiken a címet kapjuk. Ha a címkiosztást mi szeretnénk elérhetővé tenni az ügyfelek számára, a ***Enable DHCP server on interface*** négyzetet kell bejelölnünk, és hasonló módon megadnunk a csatolófelület nevét.



6. kép Rendezett, szép megjelenítés

Mint a fentiekből is látszik, ha tanult felhasználóinkat el akarjuk kápráztatni, kitűnő eszköz a Guarddog. Hasonló programokból nincs hiány, de ez a program tűnik a legalkalmasabb eszköznek, mert a felhasználók könnyen beállíthatják, és mi is könnyedén ellenőrizhetjük a beállításokat, ha a felhasználó elakad.

A versenytárs „termékek"

Nem a Guarddog azonban az egyetlen grafikus beállító csoda a szabad programok „piacán". Létezik még használható program erre a célra. Nézzük át, hogy melyek is ezek, és milyen előnyökkel, illetve hátrányokkal bírnak!

Firewall Builder

Második számú kedvencem, ha a felhasználókat arról kell meggyőznöm, hogy itt is van személyes tűzfal. Nagy előnye, hogy nemcsak a Linux csomagszűrőjét támogatja, hanem az `ipfilter`-t és az OpenBSD `pf`-jét is ismeri. A kezelőfelület GTK-ban lett megírva, ennek köszönhetően minden további nélkül fut Gnome és KDE alatt is. Felhasználási szerződése természetesen GPL. Másik nagy előnye, hogy a Debian-, illetve a Mandrake-terjesztés ezt a programot alapból tartalmazza. A beállításokat a formátumfüggetlenség jegyében XML-ben tárolja, a grafikus felület pedig a húzd és dobd szemlélet jegyében működik. A beállított szabályokat gyönyörű kivitelezésben láthatjuk.
➲ http://www.fwbuilder.org

gShieldConf

Ha nem a 2.4-es rendszermagot akarjuk használni, vagy azt használjuk, de valamilyen oknál fogva ragaszkodunk az IP Chainshez (a 2.2-es rendszermag csomagszűrője), akkor is létezik megoldás. Ez a gShield, illetve a gShiledConf, mely szintén moduláris felépítést alkalmaz – külön, saját láncok a szűréshez, feladattól vagy meghatározástól függő elnevezésekkel.
➲ http://members.shaw.ca/vhodges/gshieldconf.html

Deim Ágoston (ago@lsc.hu)
Kedveli a sört, szereti a futást és imádja Szabó Lőrinc verseit. Nem hisz vakon egyik rendszerben sem. Vonzódik a BSD-hez is. Tagja az LME-nek és a MBE-nek. Mottója: a gép nem lehet fontosabb az embernél.

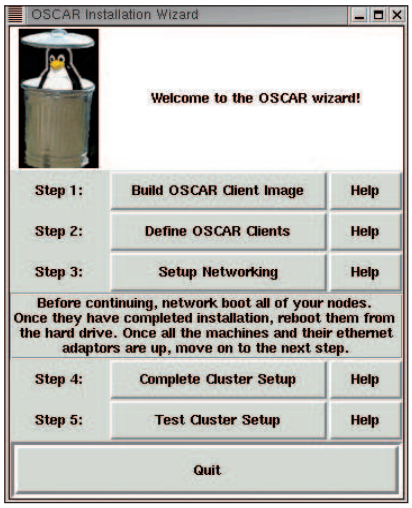
Az OSCAR-forradalom

Az Open Source Cluster Application Resource (OSCAR) története és céljai.

Tavalyelőtt áprilisban rendezték meg az első OSCAR értekezletet az Oak Ridge National Labtől egy kőhajításnyira található szállodában. Vegyes közönség gyűlt össze, képviseltették magukat az állami kutatóintézetek, az egyetemek és az ipar. Az összejövetel célja az volt, hogy az abban az időben CCDK-nak (Community Cluster Development Kit) nevezett, később az OCG-vé (Open Cluster Group) alakuló csoportról és első projektjükről, az OSCAR-ról beszélgessenek.

A résztvevők a feladatot összetevőire bontották, és minden összetevőhöz „cárt" (vezért) és „nyafogókat" (érdekelt feleket) választottak. A cárok feladata az adott összetevőhöz tartozó csoport vezetése volt, a nyafogóknak pedig elég gyakran és hangosan kellett nyafogniuk, hogy a dolgok menetrend szerint valósuljanak meg a csoport igényeinek megfelelően. Már az első találkozón, ahol megválasztották a cárokat és a nyafogókat, látszott, hogy az OSCAR fejlesztése minden eddigi program fejlesztésétől eltérő lesz. Végül is hol találhatnánk egy olyan másik projektet, amelyben az IBM, a Dell, az SGI és az Intel szorosan együttműködve dolgozik egy nyílt forrású megoldáson egy olyan óriási érdeklődés övezte témában, mint a géptelepek?

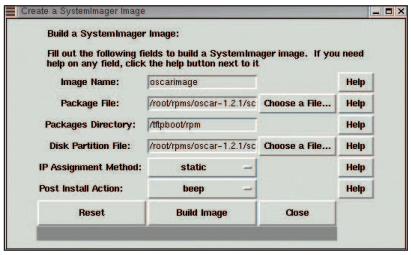
Az OSCAR ötlete vacsora közben jutott eszébe az Intelnél kutató ***Dr. Timothy Mattson***-nak és az Oak Ridge National Labnél kutató ***Dr. Stephen Scott***-nak. Éppen egy a DOE által szponzorált géptelepekkel foglalkozó értekezleten vettek részt az Argonne National Labnél, és az foglalkoztatta őket, hogyan lehetne a linuxos géptelepeket a nagyközönséggel elfogadtatni. Rájöttek, az a gond, hogy a nem programozók számára túlságosan nehéz a saját géptelep összeállításának feladata. A ***How to Build a Beowulfhoz*** (Sterling et. al.) hasonló könyvek sokat segíthetnek a számítástechnikához értőknek az alapelvek megértésében és az első géptelep megépítésében, azonban maradtak még nyomasztó gondok. Rengeteg kódot kell letölteni, amelyek megbízhatósága, támogatottsága, összeférhetősége és leírása nem egyforma. Bizonyos csomagok leírása elavult vagy ellentmondásos adatokat tartalmazott. Számos Linux-terjesztés közül lehetett választani, és mindegyik úgy különböztette meg magát a másiktól, hogy egy kicsit másképp működtek benne a dolgok. Ez azt jelentette, hogy bizonyos parancsok működése eltért egymástól, vagy más csomagokat kellett telepíteni a szolgáltatás megfelelő működéséhez. Rámutattak, hogy a gond abból adódik – mivel mindenki a saját géptelepét próbálta felépíteni azért, hogy rátegye a kezét az olcsó géptelekkel végzett számítások piacára –, hogy minden géptelepet a nulláról építettek fel. Kell lennie bizonyos gazdaságosságnak, egy helyre kell gyűjteni a legjobb programokat, eljárásokat és leírásokat, egységesíteni a különböző géptípusokra szánt csomagokat, és mindezt ingyenesen elérhetővé kell tenni. Az OSCAR projekt alapelve, hogy nem programozók is könnyen hozhassanak létre géptelepeket.



1. kép Az OSCAR-varázsló

Az első találkozó

Az Oak Ridge-i első értekezlet részvevői a következők voltak: Tim Mattson és Stephen Scott, az OCG vezetői; ***Gabriel Bonner*** az SGI-tól; ***Dave Lombard*** az MSC.Software-től; ***Rob Pennington*** az NCSA-től; ***Greg Lindahl***, aki jelenleg a Conservative Computersnél dolgozik; ***Ken Briskey*** és én az IBM-től; ***Greg Astfalk*** a HP-től; végül ***Clay Taylor*** az MPI Software Technologiestől. Nem sokkal az első értekezlet után a Veridiantól ***Broahn Mann*** csatlakozott a csapathoz, magával hozva a párhuzamos ütemezésben szerzett tapasztalatait. Csatlakozott továbbá ***Jeremy Enos*** és ***Neil Gorsuch*** az NCSA-től (akik az SSH-t valósították meg az OSCAR-ban), illetve ***Mike Brim*** az Oak Ridge National Labtől (aki a legtöbb egységesítő parancsfájlt írta, és a csomagolásban vesz részt). Az utóbbi időben csatlakozott az OSCAR projekthez ***Jeff Squyres*** és ***Brian Barrett*** az Indiana Egyetemről a LAM/MPI képviseletében. A sokszínű csoport három alapvető pontban egyezett meg:



2. kép SystemImager-lenyomat készítése

1. A géptelepek nagyteljesítményű számításokban való széles körű elterjedését gátolja, hogy nincsenek olyan elfogadott programcsomagok, amelyek elég megbízhatóak, és az átlagos felhasználók is elég könnyen használatba vehetik őket.
2. A csoport a programok terjesztésénél a nyílt forrású modellt választja. Minden hozzájárulást ingyenesen kell nyilvánosságra hozni – lehetőleg forráskód formájában és a Berkeley nyílt forrású felhasználási szerződés pontjaival összhangban.
3. A csoport céljai úgy valósíthatók meg a legjobban, ha a géptelepek tudományának úttörői által sok év kemény munkájával felhalmozott tudásra építve a legismertebb módszereket terjesztik.

Miután ezeket az alapelveket kőbe vésték, a csoport a géptelepeket alkotó összetevőket az „oszd meg és uralkodj"

elvét követve szedte össze. Az egyes összetevőkért felelős csoportok eldöntötték, hogy melyek az egyes összetevőkhöz a legjobban ismert nyílt forrású megoldások, majd az adatokat az egész csoportnak átadták. Az egyes összetevőkre kidolgozott legjobb megoldások együttesen elfogadható megoldást kínáltak az egész géptelepre. Az összetevők kidolgozása után is fáradságos és időigényes munka volt az egész rendszer összeépítése, ezt az Oak Bridge National Labben végezték Mike Brim és ***Brian Luethke*** vezetésével. Tőlük függetlenül dolgozott a Dell tesztelőcsapata ***Jenwei Hsieh***, ***Tau Leng*** és ***Yung-Chin Fang*** vezetésével. Erőfeszítéseiknek köszönhetően a személyes találkozásokon alapuló és a távolból végzett összeépítési bulikon az OSCAR olyasmivé vált, amit már érdemes volt megosztani a közösséggel.

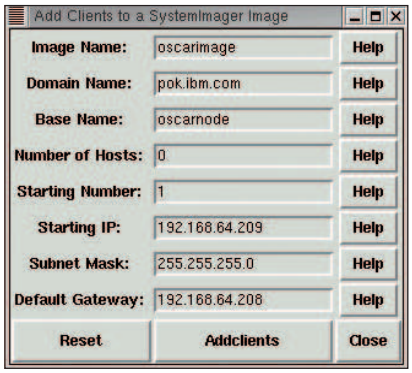
A programcsomag

Közel egy évet vett igénybe, míg az OSCAR próbaváltozatát Dallasban 2000 novemberében bemutathatták az SC2000 kiállításon, az Oak Ridge National Lab standján. A próbaváltozat eltérő kiszolgálótelepen futott, amelyet a Dell és az SGI szállított. Röviddel ezután bejelentették az első kiadást, és sikeresen szerepeltek vele a 2001 februárjában megrendezett New York-i LinuxWorld Expón az Intel standján. Azóta is folyamatosan fejlesztik az OSCAR programcsomagját, amely jelenleg az alábbi összetevőkből áll:

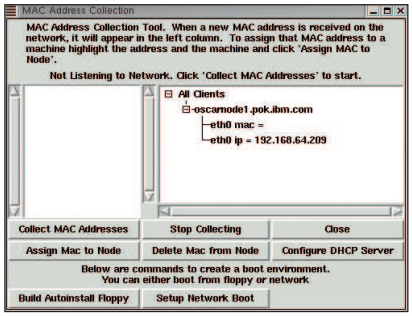
- ***Linux-telepítés***: SIS (rendszertelepítő csomag)
  A SIS egy nyílt forrású géptelep-telepítő eszköz, amely a LUI (linuxos segédprogram a géptelep telepítéséhez) és a népszerű SystemImager egyesítésével jött létre. A SIS-t az IBM-es ***Michael Chase-Salerno*** és ***Sean Dague*** fejlesztette, és az 1.2.1-es OSCAR-változatban jelent meg. Az utóbbi időben a SystemImager készítője, ***Brian Finley*** a Bald Guy Softwaretől is látogatja az OSCAR értekezleteket, és ráhajt az ingyen sörre.
- ***Biztonság:*** OpenSSH
  Ez a biztonságos kapcsolatok létrehozásának leggyakoribb módja linuxos környezetben. Az OpenSSH programcsomag kezeli a biztonságos kapcsolatokat, a kiszolgálóoldali SSH-szolgáltatásokat, a biztonsági kulcsok létrehozását és sok más egyebet, amire a számítógépek közötti biztonságos kapcsolathoz szükség van.
- ***A géptelep karbantartása:*** az egész géptelepre kiterjedő karbantartó műveletekre az OSCAR a Cluster Command and Control (C3) nevű csomagot használja, amelyet az Oak Ridge National Labnél dolgozó ***Stephen Scott*** és az East Tennessee állami egyetemen tanuló ***Brian Luethke*** fejlesztette ki. Az C3 olyan benyomást kelt, mintha egyetlen rendszeren dolgoznánk, egy parancs az egész géptelepre hatással van. A C3 telepítve marad a géptelep csomópontjain, hogy a felhasználók és a rendszergazdák később is használhassák.
- ***Programozó környezet:*** Message-Passing Interface (MPI – üzenetküldő felület) és Parallel Virtual Machine (PVM – párhuzamos virtuális gép). A géptelepek felhasználói általában maguk írják a géptelepen futtatandó programokat. Több különböző megközelítés létezik a programírásra e környezetben. A leggyakoribb módszer az üzenetküldő könyvtárak használata. Jelenleg az OSCAR által telepített fordítóprogramok és a matematikai könyvtárak a Linux-terjesztésből jönnek.
  A LAM/MPI és az MPICH is elérhető az OSCAR 1.1 óta.
- ***Terheléselosztás:*** Portable Batch System (PBS) a Veridiantől és a Maui Scheduler (a Maui High Times Computing Center fejlesztése).
  A géptelep idejének beosztásához valamilyen terhelés- és feladatelosztás szükséges. A Maui az OSCAR feladatelosztója – ez osztja el az erőforrásokat és dönt az ütemezésről. A PBS a feladatkiszolgáló, illetve -indító, a feladatok indításán és leállításán kívül a feladatsorokat is kezeli.

Az MSC.Software és az OSCAR

Az MSC.Software Corporation Systems divíziója által kifejlesztett MSC.Linux-terjesztés különösen fontos az OSCAR elfogadtatása szempontjából. Röviddel az OSCAR 1.0-s változatának megjelenése után az MSC.Software bejelentette saját megoldását a géptelepekre, ez volt az MSC.Linux Version 2001 operációs rendszer. Ez a 2001-es ajánlat nagyrészt az OSCAR-on alapult, és az első olyan kereskedelmi termék volt, amely az OCG munkája nyomán jött létre. ***Joe Griffin*** az MSC.Software-től Webmin felületet adott a LUI-hoz (a LUI az első OSCAR géptelep-telepítő eszköz volt), amely alacsony szintű LUI-parancsokat készített a csomópontok számára. Ennek a felületnek a segítségével könnyen



3. kép Ügyfelek hozzáadása a SystemImager-lenyomathoz



4. kép MAC-címek összegyűjtése a SystemImager-lenyomathoz

megadhatók a géptelep csomópontjai és az egyes csomópontokra telepítendő erőforrások. Az OCG egyik eredeti célja az volt, hogy a vállalatok felismerjék a nyílt OSCAR programcsomag értékét, és saját zárt vagy nyílt programcsomagot építsenek az OSCAR köré. Ez esetben ugyanis megmenekülnek a géptelep felépítésének aprólékos munkájától, nem nekik kell az alapvető szerkezeteket kitalálni, és sokkal jobban összpontosíthatnak azokra a világszínvonalú fejlesztésekre, amelyekkel a saját megoldásukat megkülönböztethetik a többiekétől.

A közös munka

Mint más szerteágazó nyílt forrású projektnél, itt is a kezdetektől világos volt, hogy a társaság nem feltétlenül tud egy időben és egy helyen együtt dolgozni. Az utazási költségek egyszerűen túl magasak voltak, és ilyen sok ember idejének egyeztetése is nehézséget jelentett. A munka irányítására a csoport hetente telefonértekezletet tartott, használták a levelezőlistát, és tanácskozások vagy kiállítások alkalmával néha összejöttek. Szemtől szemben a fejlesztők a negyedévente tartott „összekovácsolási" bulikon találkoztak, az egyik ilyen az Intelnél volt az oregoni Hillsboróban, a másik az NCSA-nél Illinoisban. Hogy az egység az összejövetelek között is megmarad-

jon, bevezették az úgynevezett elosztott egységbe rendező buli intézményét, angol rövidítéssel a DIP napot. A DIP napokon minden résztvevő, akinek van géptelepe, idejét az OSCAR-nak szenteli. Mindenki letölti az OSCAR csomagjait, telepíti, futtatja, és a hibákat jelenti a csoportnak. A DIP napokon a programozóktól elvárják, hogy a hibákat azonnal javítsák, így a kódot rövid időn belül egymás után sokszor ki tudják próbálni. A DIP nap alatt több konferenciabeszélgetésre is sor kerül, megtárgyalják az eredményeket, új feladatokat tűznek ki, és meghatározzák a feladatok fontossági sorrendjét. A DIP napokkal és a személyes találkozásokkal irányított OSCAR projekt nagy előrehaladást tett a megbízhatóság és a tudás terén.

Az első benyomások az OSCAR-ról

Az első dolog, amit az OSCAR fájljának kicsomagolása után észrevehetünk, hogy készítői alapos munkát végeztek. Kiterjedt leírás szól a telepítésről, a rendszerkövetelményekről, a felhasználói szerződésről (GPL) és az OSCAR mögött meghúzódó elméletről. Létezik gyorstalpaló tanfolyam a türelmetleneknek és egy teljes leírás is. Azt is megfigyelhetjük, hogy semmi egyebet nem kell letölteni, minden benne van az OSCAR egyetlen *tar* fájljában. Az OSCAR a géptelepek hagyományos szemléletén alapul: egy kiszolgáló és N számítócsomópont van. A kiszolgáló felelős a csomópontok telepítéséért, ütemezéséért és megfigyeléséért. A géptelep csomópontjainak egyformáknak kell lenniük, azaz ugyanazt a Linux-terjesztést és ugyanazt a változatot kell használniuk. A felhasználó először az `install_cluster` parancsot adja ki, ami sok mindent elvégez. Létrehozza a szükséges könyvtárakat, kezeli az NFS-t és az `xinetd`-t, telepíti a LAM/MPI-t, a C3-at, a PBS-t, a Mauit, az OpenSSH-t, a SIS-t, a Perlt, a SystemImagert és az MPICH-t, frissíti a különböző profilokat és beállító parancsfájlokat, végül elindítja az OSCAR-varázslót.

Ha minden jól megy, kellemes meglepetés ér bennünket: ez az OSCAR-varázsló. Az OSCAR-csapat úgy érezte, hogy a varázsló az OSCAR egyik megkülönböztető jegye lesz a linuxos géptelepek megoldásai között. A varázsló célja egyértelmű – követni kell az utasításokat, és a géptelep fájdalommentesen telepíthető. A varázsló minden lépésének van egy belépési és egy kilépési feltétele. Ha a kilépési feltétel teljesül, az OSCAR egy sikerüzenetet ad vissza, ami azt jelzi, hogy biztonságosan a következő lépésre lehet ugrani. A varázslót követve a ***Build OSCAR client image*** gomb megnyomásával a második panelre érünk, neve ***Create a SystemImager Image*** panel.

A ***SystemImager*** panel célja, hogy olyan fájlrendszert hozzon létre a kiszolgálón, amely később minden ügyfélre telepítve lesz. Az ***Image Name*** mező lehetővé teszi, hogy a felhasználó több ***SystemImager*** lenyomatot készítsen, mindegyiket egyedi névvel ellátva. A ***Package File*** mezőben adhatjuk meg, hogy milyen csomagokat telepítünk az ügyfelekre. Az OSCAR által nyújtott alapbeállítás a legtöbb felhasználó igényeit kielégíti. A ***Packages Directory*** mezőben adhatjuk meg az RPM-csomagok forrását, végül a ***Disk Partition File*** mezőben lehet testreszabni a lemez felosztását. Az OSCAR erre is kínál alapbeállításokat, IDE- és SCSI-meghajtókra egyaránt. A ***Build Image*** gomb megnyomása után megindul az ügyféllenyomat készítése a kiszolgálón. Ha ez kész, továbbléphetünk a varázsló második pontjára, az OSCAR-ügyfelek megadásához.

Az ***Add Clients*** panelen a felhasználó megadhatja az új ügyfelekhez rendelendő IP-címtartományt. Minden egyes ügyfélhez egy lenyomatnév tartozik, amelyet az ***Image Name*** mezőbe kell írni. Egy ügyfélcsoportot IP-címtartomány formájában adhatunk meg, hogy mindegyik ügyfélnek azonos legyen az alhálózati maszkja és az alapértelmezett átjárója. Az ***Addclients*** gomb az ügyfél beállítófájlját készíti el a SIS számára. Ha ez megvan, lépjünk tovább a varázsló harmadik pontjára, a hálózati beállításokhoz.

A ***Setup Networking*** panelen a géptelep csomópontjainak MAC-címeit gyűjtjük össze. Ha a csomópont a hálózatról rendszert tud indítani (PXE), egyszerűen rendeljük hozzá a MAC-címhez az ügyfelet, és kapcsoljuk be. Ha csomópont nem ismeri a PXE-t, készíteni kell egy ***SystemImager*** rendszerindító lemezt a ***Build Autoinstall Floppy*** gomb segítségével.

A MAC-címek összegyűjtése után nyomjuk meg a ***Configure the DHCP*** gombot, és kapcsoljuk be a csomópontokat, hogy megkezdődjön a Linux telepítése.

A csomópontok telepítése után minden csomópont elkezd idegesítően és kitartóan sípolni, jelezve, hogy el kell távolítani a rendszerindító lemezt vagy ki kell kapcsolni a PXE-t, és a csomópontot újraindítani a merevlemezről. Miután mindegyik újraindult, a csomópontok készen állnak a ***Complete Cluster Setup*** gomb megnyomására. Ez a művelet egyezteti az időt és lefuttat néhány csomagfüggő telepítés utáni parancsfájlt.

A ***Test Cluster Setup*** gomb hatására rövid feladatok futnak le – kipróbálva az ütemezőt és a párhuzamos könyvtárakat. Ha a géptelep telepítése elkészült és minden működik, a próbaparancsfájlokat futtathatjuk, melyek a géptelep általános egészségi állapotát mérik fel.

A `test_install` parancsfájl ellenőrzi, hogy a PBS és a Maui ütemező be van-e állítva és fut-e, a C3 eszközök telepítve vannak-e, és a géptelep készen áll-e párhuzamos feladatok végrehajtására.

Az OSCAR jövője

A cikk írásának idején az OSCAR 1.2.1 a legfrissebb, amely Red Hat Linux 7.1 alatt fut. Az MSC.Linux Version 2001 az OSCAR 1.1-en alapul. Az 1.0 és 1.1 kiadá-

Kapcsolódó címek

A LUI projekt honlapja ➲ http://oss.software.ibm.com/lui
Az MSC.Linux terjesztés letöltési helye ➲ http://www.msclinux.com
OCG- és OSCAR-előadások és -cikkek ➲ http://www.csm.ornl.gov/oscar
Az OCG honlapja ➲ http://www.openclustergroup.org
Az OSCAR projekt honlapja ➲ http://www.sourceforge.net/projects/oscar
A System Configurator projekt honlapja
➲ http://www.sourceforge.net/projects/systemconfig
A SystemImager projekt honlapja
➲ http://www.sourceforge.net/projects/systemimager
A System Installation Suite projekt honlapja
➲ http://www.sourceforge.net/projects/sisuite
A SystemInstaller projekt honlapja
➲ http://www.sourceforge.net/projects/systeminstaller
A Tennessee Oak Ridge Cluster projekt honlapja
➲ http://www.epm.ornl.gov/torc/TORChomepage.htm
A TORC projekt honlapja
➲ http://www.epm.ornl.gov/torc/TORCprojectspage.htm

sok eléggé népszerűek voltak a közösség tagjai között – körülbelül 25 000-szer töltötték le a SourceForge-ról. Jelenleg az az OSCAR legnagyobb gondja, hogy viszonylag kevés fejlesztőjének időt és energiát kell áldoznia az olyan új programcsomagok beépítésére, amelyeket az emberek az OSCAR-ban látni szeretnének. Az OSCAR 2.0 projekt már jó úton halad, és ennek a változatnak az egyik hangsúlyos pontja, hogy az összetevőket szabványos API-kon keresztül lehessen bővíteni, így bárki hozzáteheti majd a nyílt forrású csomagját az OSCAR-hoz. Maga az OCG is nő. Amióta Tim Mattson az Inteltől egyéves szabadságra távozott, Jeff Squyres az Indiana Egyetemről vette át a vezetést az OSCAR 2.0 felépítése és egységessége témakörében. ***Ibrahim Haddad*** az Ericcsontól (sokat ír a Linux Journalba is) olyan érdekes ötletekkel érkezett a társasághoz, amelyek lehetővé tennék, hogy az OSCAR közel olyan megbízható legyen, mint a telekommunikációs alkalmazások. ***Jim Garlick*** a Lawrence Livermore National Lab képviseletében szintén csatlakozott a társasághoz, magával hozva a nagy géptelepek méretezéséről szerzett tapasztalatait.

Felületek és terjesztések

A legelső OSCAR értekezleten a felek megegyeztek, hogy az OSCAR egyetlen Linux-terjesztéstől vagy felülettől sem függhet. Ennek ellenére az OCG mostanáig főleg a Red Hat és az MSC.Linux terjesztésekre összpontosított az IA-32 felületen. 2002-ben a látókör tágulni fog. A SIS OSCAR-ba építésének az volt a célja, hogy minden RPM-en alapuló terjesztést támogathassanak, ezek a SuSE, Turbolinux, Red Hat, MSC.Linux és a Caldera. A későbbiekben a deb-eken alapuló terjesztések támogatása is megvalósul (pl.: Debian). Ráadásul a SIS felépítése lehetővé teszi, hogy más felületekre is könnyen átvigyék. Az NCSA máris rendelkezik egy Itaniumon futó OSCAR próbával, és az Oak Ridge a Red Hat 7.2-t próbálja ki. Mivel az API nyílt, és az OSCAR számos terjesztés alatt és nagyszámú felületen képes futni, ebben az évben nagy fellendülés várható.

Záró gondolatok

Az OSCAR hatása a linuxos géptelepeket használó közösségre több szemszögből is vizsgálható. A legnyilvánvalóbb előny, hogy az OSCAR hasznos segédeszköz a géptelep működtetéséhez, és a legkülönfélébb gyártók rendszerein is használható. Sok nehézséget elkerülhetünk a segítségével, mert a szükséges programokat nem kell különböző webhelyekről összeszedegetni. Ez tényleg olyan megoldás géptelepek létrehozására, amelyet a nem programozó felhasználó is képes használni. A felszín alatt az OCG egyre növekvő szervezetét találjuk, amelyben részt vesznek az állami kutatóintézetek, az egyetemek és az ipar képviselői, hogy együtt új nyílt forrású megoldást fejlesszenek Linuxra. Visszatekintve számára a talán legfontosabb hosszú távú hozzájárulás a közösség az, hogy a nyílt forrás jobbá tétele érdekében a közös munkálkodás új formáit fejlesztették ki.

Linux Journal 2002. június, 98. szám

Richard Ferri
az IBM Linux Technology Centerben vezetőprogramozó. Nyílt forrású linuxos géptelepeken dolgozik.

Linux-parancsok: szerelem első látásra

Amikor parancsokról beszélünk, ez minden alkalommal a parancsfelületen való munkát jelenti. Ez az a bizonyos parancsjel ($), ami nagyon sok parancshéjnál közös. Amikor rendszergazdaként jelentkezünk a rendszerbe, jobbára eltérő parancsjelet (prompt) láthatunk. Az ilyenkor használatos jelnek (#) igen sok elnevezése ismeretes: Észak-Amerikában „pound sign” (fontjel) vagy „hash mark” a neve, angol barátaim azt mondják, ez egy „octothorp”, mások tic-tac-toe táblának nevezik (magyarul pedig kettős kereszt vagy rács a neve – a ford.). Én parancsjelnek (`root prompt`-nak) fogom nevezni. Végül is nem az elnevezés fontos, csak egy olyan parancsjelet szeretnénk, amivel elkezdhetjük a gyakorlást. Ha grafikus felületen vagyunk, kattintsunk a terminálablak ikonra, hogy elinduljon a terminál- (vagy héj-) program. A KDE-felhasználók egy Konsole-t indítanak, míg a Gnome-felhasználók valószínűleg egy színes `xterm`-et. Ha a terminálemulátorok számát találomra próbálnánk megbecsülni, nagy valószínűséggel kevesebbre gondolnánk, mint ahányféle valójában létezik. Használható a tiszteletre méltó Xterm, a Konsole, az rxvt és Eterm, hogy csupán néhányat említsek.

Parancsok, amelyeket szeretni érdemes

`date`	Dátum és idő megjelenítése.
`who`	Az adott pillanatban bejelentkezett felhasználók.
`whoami`	Ki is vagyok valójában?
`tty`	Az adott munkaállomást azonosítja.
`echo`	Hello, ello, llo, lo, o, o, o … (visszhang).
`finger`	Egy felhasználót azonosít. Megállapít róluk néhány dolgot.
`last`	Ki jelentkezett be utoljára és még mindig bent van-e?

Állománykezelés

Hadd áruljam el a számítógépek, operációs rendszerek és az ezeket körülvevő egész iparág nagy titkát: minden dolog adat, és minden, amit a számítógépekkel teszünk, legvégső célnak az ebből nyerhető információt tekinti. A fájlok tulajdonképpen raktárai ezeknek az információknak. Az adatok ügyes kezelése, karbantartása, élni és visszaélni velük – még húsz év múlva is ez lesz a számítástechnika központi kérdése. A következő dolog, amiről beszélni szeretnék, az a három fájl, ami felett a legtöbbször siklik át a tekintetünk: a szabványos bemenet, a szabványos kimenet és a szabványos hiba. E „fájlok” kezelésében szerzett jártasság bámulatos rugalmasságot fog biztosítani, amikor mindennapi munkánkat kell majd végeznünk.

`ls`	Állományok listázása (LiSt)
`cat`	Állományok összefűzése (conCATenate).
`sort`	Egy állomány (vagy kimenet) tartalmának rendezése (SORT).
`uniq`	Rendezés után az egyedi (UNIQue) sorokkal tér vissza.
`wc`	Szavak megszámlálása (Word Count). A sorok, szavak és karakterek számát adja eredményül.
`cp`	Fájlok másolása (CoPy).
`mv`	Fájlok mozgatása (MoVe) vagy átnevezése.
`rm`	Egy állomány eltávolítása (ReMove) vagy törlése.
`more`	Lehetővé teszi nagy szöveges állományok könnyű lapozását.
`less`	Mint a more parancs, de komolyabb hozzáállással.

Részlet Marcel Gagné: Linux-rendszerfelügyelet című könyvéből

Az X beállítása (1. rész)

Az alapoktól egészen több monitor használatáig.

A két részből álló leírás első része inkább a kezdő felhasználóknak szól, akik az X beállításában még nálam is kevesebb jártassággal rendelkeznek. Elsősorban az egyetlen működő X-felhasználóval rendelkező rendszer beállításaival ismerkedünk meg.

Az X indítása

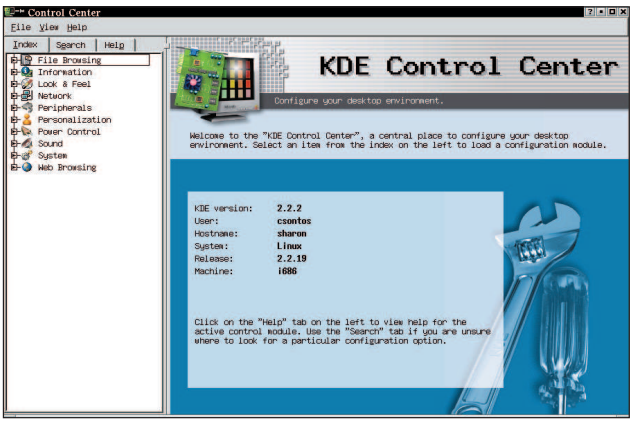
A beállítások ismertetése előtt alapszinten meg kell ismernünk az X működését. A grafikus felületet valamelyik grafikus kiszolgálónak (X-kiszolgálónak szoktuk nevezni) a feladata szolgáltatni. Ez a kiszolgáló csak a grafikus felület alapját szolgáltatja, ami a monitoron szürkés színű alapfelületet ad, ezután a felhasználónak (pontosabban az általa módosított indítófájlnak) a feladata az ablakkezelő indítása. Az X-kiszolgálót nem szabad összetéveszteni a grafikus felületkezelővel. Mint említettem, az X szolgáltatja a grafikus felület alapját, amihez a beállított indítási folyamattól függően kapcsolódik, illetve kapcsolódhat a Grafikus felületkezelő program, és ez egy hétköznapi felhasználó számára is könnyen használható felületet ad. A Grafikus felületkezelőnek csak a grafikus felületen történő bejelentkezések kezelésében van szerepe. Ezen eszközök önmagukban nem az egyes felhasználó által munkához, játékhoz használható „munkaasztalt” adják, ezért valamilyen módon az ablakkezelőt is el kell indítanunk (az ablakkezelőkről a Linuxvilág előző néhány számában esett szó). Az ablakkezelők indításának tehát két alapvető módja van, lássunk ezekre egy-egy velős példát.

A KDM

A felhasználóbarátabb Linux-rendszerek többsége operációs rendszerünket általában alaphelyzetből úgy akarja beállítani, hogy a telepítést követő első rendszerindításkor a felhasználó azonnal a grafikus felülettel találkozzon. Ez azt jelenti, hogy a rendszerindítási folyamat utolsó mozzanataként elindítja valamelyik Grafikus felületkezelőt (Desktop Managert). A legelterjedtebbek ezek közül az XDM, GDM és a KDM. Most röviden a KDM-ről és beállítóeszközéről írnék, mivel talán ez az az eszköz, ami grafikus felületen is a legkönnyebben állítható be. A *képen* látható beállítóeszköz a KDE ***KControl*** nevű eszközli. Segítségével beállíthatjuk, hogy a számítógép indításakor egy adott felhasználóönműködően bejentkezzen. Ha nincs ilyen felhasználó megadva, egy felhasználónév, jelszó és a kívánt grafikus felület kiválasztását lehetővé tevő ablakkal találkozhatunk – megjelenése (háttér, betűtípusok, a felhasználók ikonjai, a látható felhasználói ikonok stb.) a KControlban mind-mind beállíthatók. Amennyiben a belépő felhasználó kiválaszt egy ablakkezelőt, a KDM legközelebbi belépésekor is ezt fogja felajánlani számára.

Az xinitrc fájl

Az ablakkezelők a felületkezelők nélkül is indíthatók. Ehhez előbb konzolon be kell jelentkeznünk, majd parancssorból indíthatjuk a grafikus felületet. Erre a célra több parancs is használható (`startx`, `xinit`). Beállításukat – vagyis azt, hogy pontosan mi is induljon el az X indításakor – az ***xinitrc*** fájlban tehetjük meg. Az alábbi sorokban az ***xinitrc*** fájl tartalmából láthatunk egy részletet:




```
#!/bin/sh
xterm -fn 7x13bold -g 80x32+10+50 &
xterm -fn 9x15bold -g 80x34+30-10 &
xscreensaver-command -exit &
xscreensaver &
exec fvwm2
```

Mint megfigyelhető, a grafikus ügyfél különböző helyzetben két `xterm` ablakot nyit meg, ezt követően az `xscreensaver` képernyőkímélőt, végül az `fvwm2` ablakkezelőt.
Az általunk kiadott parancsok után az `&` (és-jel) használata célszerű, mert a parancsértelmező ekkor nem várja meg míg az elindított program befejezi futását.

A grafikus eszközök beállítása

Először is tisztáznunk kell, hogy videokártyánkat az X támogatja-e. A Matrox és az nVidia kártyák talán a legelterjedtebbek ezek a támogatott eszközök közé tartoznak. Ezekkel általában csak akkor lehet gondunk, ha X-ünk túl régi vagy kártyánk nagyon új, és még nincs hozzá megfelelő meghajtóprogram. Én most egy Matrox Millennium G550 kártyát használok, előtte egy ASUS V3400-as Riva TNT kártyám volt. Mindkettő AGP-s eszköz és egyikkel sem volt igazán gondom. E két kártya esetében az X tartalmazta a megjelenítéshez szükséges meghajtóprogramot, de a jobb képminőség elérése érdekében a meghajtóprogramokat mindegyiknél frissítettem. Az nVidia kártyánál a külső eszközmeghajtó program esetében fontos volt megtenni két beállítást a BIOS-ban. Az egyik a plug and play operációs rendszer lehetőség kikapcsolása, a másik a VGA IRQ igenre állítása. A meghajtóprogramok telepítése egyik kártya esetében sem bizonyult bonyolult feladatnak, hiszen – mint sok esetben – a gyárilag elkészített csomagokat használhatjuk (a SuSE nVidia-meghajtó RPM-je). Az sem jelent gondot, ha csak a forrás áll rendelkezésünkre, ekkor is csupán annyi a feladatunk, hogy a kicsomagolt eszközmeghajtó könyvtárában rendszergazdaként lefuttatjuk a `make` vagy a `make install` parancsot. Matrox-kártyák esetében a támogatottság erősebb, hiszen a rendszermagban is számos helyen találkozhatunk kapcsolódó beállítási lehetőségekkel.
Az X beállításához a következő parancsokkal ugyancsak hasz-

nos tájékoztatást kaphatunk (fontos, hogy használatukkor semmilyen grafikus felület vagy felületkezelő ne működjön):
`-configure`: kipróbálja az eszközöket, és beírja őket az XF86Config fájlba.
`-probeonly`: kipróbálja az eszközöket, majd kilép.
`-scanpci`: megállapítja, hogy melyik PCI-, AGP-foglalatban van valamilyen grafikus kártya, és kiírja a több monitor használatakor szükséges PCI-értékeket.
Sorozatunkban csak a 4.x-es X-változatokkal foglalkozom, mivel ez a jövő iránya, és a régi grafikus eszközökhöz is egyre több meghajtóprogram érhető el benne.

Az XF86Config-4 fájl beállítása

A 4.x-es X grafikus felületének alapbeállításait az ***XF86Config-4*** fájl tartalmazza, amit az esetek igen nagy részében a */etc/X11/* könyvtárban találunk meg. A fájl elkészítésére több lehetőségünk nyílik. A fájlt Red Hat alatt az Xconfigurator (utódja újabban a setupból érhető el), Madrake alatt az Xdrakres, SuSE alatt pedig – mivel ezt nagyon rég próbáltam már – legszebb emlékeim szerint a Yastból lehetett elkészíteni és módosítani.
Debian alatt az X telepítésekor karakteres (xf86config) felületen vagy grafikus (xf86cfg) nyílik lehetőségünk elvégezni a szükséges beállításokat, vagy a telepítés után a `dpkg-reconfigure xserver-common`-nal, és a `dpkg-reconfigure xserver-xfree86`-tal parancssorból. Egy mintafájlt láthatunk itt:
`/usr/X11R6/lib/X11/XF86Config.eg`
Az alábbi rész nagyrészt az ***XF86Config-4*** súgóoldaára épül – a teljesség igénye nélkül, hiszen csak a számomra ismerős és eddig szükségessé vált beállításokat emeltem ki belőle.
E beállítóprogramok több-kevesebb sikerrel és részletességgel beállítják és átszerkesztik az ***XF86Config-4*** fájlt.
Ha úgy érezzük, hogy a lehető legtöbbet kis szeretnénk hozni a grafikus felületből, a beállításokat mindenképpen nekünk magunknak célszerű átszerkesztenünk. Az ***XF86Config-4*** fájl a következő jól elkülönülő részeket tartalmazza:
Files: a szükséges fájlok elérési útvonala.
ServerFlags: kiszolgálóbeállítások.
Modules: dinamikusan beolvasható modulok.
InputDevice: bemeneti eszközök (egér, billentyűzet).
Device: grafikus eszközök, videokártyák.
VideoAdaptor: az Xv videoátalakító beállításai.
Monitor: monitorbeállítások.
Modes: a videomódok leírásai.
Screen: képernyőbeállítások.
ServerLayout: a beállítások összesítése.
DRI: A DRI megjelenési alrendszer beállításai.
Vendor: gyártófüggő beállítások.
Az egyes leíró részek felépítése meglehetősen egyszerű:

```
Section "Files"
...
EndSection
```

A fájlon belüli egyes részek sorrendje lényegtelen, bár a könnyebb eligazodás végett célszerű az eredeti vagy mintafájlban található sorrend betartása. Mintafájlokat a meghajtóprogramok leírásában, és az HOGYAN-ok X-re vonatkozó részeiben találhatunk. Lássuk az egyes részeket beállítási lehetőségeikkel együtt!

Files

Az X kiszolgáló számára szükséges elérési útvonalak beállítására a parancssori beállítás lehetőségeinek megfelelően (man Xserver, Xfree86) a ***Files*** rész használható. A parancssori beállítások a fájlban beállított lehetőségeket felülírják.
Ez a rész alapesetben alapvetően három dologról gondoskodik.
Az első az RGB adatbázis elérési útvonala:

```
RgbPath "/usr/X11R6/lib/X11/rgb"
```

A másik két beállítás az X alatt használható betűtípusok kiválasztására szolgál. Az első példa a betűkészlet-kiszolgáló elérési módját adja meg. Ennek leírás szerinti formátuma a zárójelek nélkül az alábbi: (kapcsolati protokoll, például TCP)/(gépnév, például linux.domain.net):(a kiszolgáló kapuszáma, ez általában 7100)
Példa az ***XF86Config-4*** fájlból.

```
FontPath "unix/:7100"
```

Ezt általában az alap betűkészletek teljes elérési útvonalának megadása követi:

```
FontPath "/usr/lib/X11/fonts/100dpi"
```

Itt az összes létező betűkészlet könyvtára megadható. Ha a betűkészlet-kiszolgáló nincs megadva, a kiszolgáló a betűkészleteket saját maga próbálja meg elérni.
Még egy elérési útvonal adható meg a ***Files*** részben, a `ModulePath`, ami az X által betöltendő modulok nevét sorolja fel. Fontos újdonság ,hogy az X4.x képes közvetlenül kezelni a TrueType (FreeType) betűtípusokat, így leegyszerűsítjük rendszerünket, ehhez a `freetype` vagy az `xtt` modulok használatoak.

ServerFlags

Ez a rész a kiszolgáló által használt beállításokat tartalmazza, az alábbi példának megfelelően:

```
Section "ServerFlags"
AllowMouseOpenFail
Option "NoTrapSignals" "true"
#(a vÆltoz  ØrtØke igaz (true), vagy hamis
#(false) lehet)
EndSection
```

Az itteni beállítások felülírják a ***ServerLayout*** részben megadott beállításokat. Az itt beállított kapcsolók parancssorból történő indítás során az ott megadott értékekkel felülírhatók. A következő tulajdonságok állíthatóak be:

- ***Option*** "***NoTrapSignals***" (true/false)
  Törli vagy nem törli a memórialenyomatot (a core fájlt), ha az X valamilyen végzetes hiba miatt összeomlik. Alapesetben erre a fájlra nincs szükségünk, csak akkor, ha az X-et fejlesztjük.
- ***Option*** "***DontZap***" (true/false)
  Ez tiltja a CTRL+ALT+BACKSPACE billentyűkombináció használatát. Alapesetben ezzel a kombinációval az X-kiszolgáló kilőhető (ha a kapcsoló értéke `true`, az X-et a leírt módon nem lehet leállítani).
- ***Option*** "***DontZoom***" (true/false)
  Engedélyezi a CTRL+ALT+Numerikus billentyűzet + (plusz), illetve a - (mínusz) billentyűkombinációk használatát. Működő X esetén ezek teszik lehetővé az engedélyezett videomódok közti váltást.
- ***Option*** "***DisableVidModeExtension***" (true/false)
  Nem engedélyezi az `xvidtune` ügyfél által használt VidMode-kiterjesztés használatát.
- Opti***on*** "***AllowNonLocalXvidtune***" (true/false)
  Az `xvidtune`-ügyfél (az, amelyik a VidMode-kiterjesztést használja) a kapcsolódást teszi lehetővé egy másik gépről.
- ***Option*** "***DisableModInDev***" (true/false)
  Nem teszi elérhetővé az XFree86-Misc-kiterjesztését, amelylyel az eszköz beállításait dinamikusan lehet módosítani.
- ***Option*** "***AllowNonLocalModInDev***" (true/false)
  Egy másik gépről kapcsolódó ügyfél számára a működő kiszolgálón lehetővé teszi a billentyűzet- és az egérbeállítások módosítását.

- *option* "***AllowMouseOpenFail***" (true/false)
  Akkor is lehetővé teszi a kiszolgáló elindítását, ha a mutatóeszköz nem indítható. Ez egyfelhasználós környezetben nem szerencsés, hiszen sok ablakkezelő az egér használatára épít.
- ***Option*** "***BlankTime***" (idő)
  Beállítja a képernyő elsötétítésének alapértékét.
- ***Option*** "***StandbyTime***" (idő)
  A monitor ***Standby*** állapotba kerülésének idejét állítja be. Csak VESA DPM-támogatással rendelkező monitorok esetében célszerű használni.
- ***Option*** "***SuspendTime***" (idő)
  A monitor ***Suspend*** állapotba kerülésének ideje.
- ***Option*** "***OffTime***" (idő)
  A monitor lekapcsolásáig eltelő idő.
- ***Option*** "***Pixmap***" (bit/pixel)
  A megjelenített kép színmélységét 24-re állítja – 24-es és 32-es színmélység esetén használható.
- ***Option*** "***NoPM***" (true/false)
  Nem engedélyezi az ***Energiatakarékos üzemmód***-okat.
- ***Option*** "***Xinerama***" (true/false)
  Engedélyezi a XINERAMA-kiterjesztést. Erről a későbbiekben még lesz szó.

Module

A ***Modulok*** rész az X által beolvasott modulokat sorolja fel. Az itt található bejegyzések két formátumban fordulhatnak elő, az első valahogy így néz ki:

```
Load "modulnev"
```

A második változat pedig így:

```
SubSection "extmod"
Option "omit XFree86-DGA"
EndSubSection
```

A modulokat a rendszer a `ModulePath` változóban megadott elérési útvonalon keresi.
Hogy megtudjuk, mely modulok tölthetők be, nézzük meg a következő könyvtárakat. A modulokat kiterjesztés nélkül a fenti példákhoz hasonlóan olvastathatjuk be:

```
/usr/X11R6/lib/modules/fonts
/usr/X11R6/lib/modules/extensions
```

A `bitmap` modul önműködően betöltődik. Néhány többször használt modul:

`Load "Glcore"` – az OpenGL hibajavításához szükséges.
`Load "glx"` – az OpenGL használatához szükséges (játékok, MPlayer).
`Load "dri"` – a DRI közvetlen leképezőrendszer engedélyezése.
`Load "pex5"` – pex5 betűtípusok kezelését végzi.
`Load "dbe"` – Dupla pufferkiterjesztés a szebb megjelenítéshez.
`Option "omit xfree86-dga"` – nem indítja el a DGA-kiterjesztést.
`Load "type1"` – A type1-es betűtípusokat raszterizálja, és megjelenésüket simítja.
`Load "freetype"` – ahogy már említettem, az új rendszer e modul segítségével képes a FreeType betűtípusok kezelésére (lásd még az új `xtt` modult).

InputDevice

Az ***InputDevice*** rész alapesetben legalább kétszer fordul elő a fájlban: egyszer a billentyűzet esetében és egyszer az egérében. Formátuma az alábbi:

```
Section "InputDevice"
Identifier "elnevezes"
Driver "meghajt  program neve"
beÆll tÆsok
...
EndSection
```

Az `elnevezes` egy egyedi név, amire később hivatkozni tudun k. A Driver az eszközhöz tartozó meghajtóprogram neve. Nézzünk egy példát:

```
Section "InputDevice"
Identifier "Mouse0"
Driver "mouse"
Option "Device" "/dev/mouse"
Option "Protocol" "ImPS/2"
Option "Emulate3Buttons" "off"
Option "ZAxisMapping" "4 5"
EndSection
```

A `Mouse0` azonosítóeszközt a ***mouse*** meghajtóprogrammal használjuk. Az eszköz a ***/dev/mouse*** eszközre kapcsolódik PS/2-es protokollal. Valószínűleg háromgombos egérről van szó, mivel a harmadik gomb emulációja ki van kapcsolva (ha be lenne, a kétgombos egér két gombjának egy idejű lenyomásával a középső egérgombhoz kapcsolódó parancsot tudnák végrehajtatni.)
A `"ZAxisMapping" "4 5"` beállítás görgős egerek esetében fontos, ahol a három gombon felül beérkező adatok feladatát meg kell adni. Érdemes, ha valakinek ilyen egere van, a 4-est felcserélni az 5-össel, mivel várhatóan azt tapasztalja majd, hogy a görgetés iránya felcserélődik.
Néhány X alatt használt egérprotokoll:
`Microsoft`: egyszerű Microsoft egér.
`MouseSystems`: alapprotokoll a háromgombos soros egerekhez.
`PS/2`: ezt a protokollt használják a busra kapcsolódó egerek (PS/2-es egerek).
`ImPS/2`: ezzel azonos az „IntelliMouse” protokollelnevezés.
A PS/2-es kapura kapcsolódó egerek használják, valamint néhány USB-s egér. Görgős egerek beállítására alkalmas protokoll.
A billentyűzet a másik eszköz, amit szintén külön ***InputDevice*** részbe célszerű tenni:

```
Section "InputDevice"
Identifier "Keyboard0" # ez lesz az eszk z
                       # kØsıbbi azonos t ja
Driver "keyboard" # a meghajt program neve
Option "XkbRules" "xfree86"
# A hasznÆlt billentyßzet t pusÆnak
# megadÆsa (/usr/X11R6/lib/X11/xkb/rules)
Option "XkbModel" "pc105"
# A billentyßzet gombjainak szÆma
# (ezzel megadjuk a gombok helyzetØt, k djÆt.)
Option "XkbLayout" "hu"
# A nyelvi beÆll tÆs k dja
# (/usr/X11R6/lib/X11/xkb/symbols)
EndSection
```

Két fontos beállítás, amit a megfelelő eszközöknél érdemes beállítani:

```
Option "CorePointer"
Option "CoreKeyboard"
```

A `Core*` eszköz lesz az alapértelmezett eszköz.
Terjedelmi korlátok miatt ebbe a részbe most sajnos ennyi fért. Következő írásomban több monitor beállításának és egyidejű használatának lehetőségeiről olvashattok.

Tóth Béla
(tothb1@freemail.hu) Nős, két gyermek büszke atyja. Egyaránt otthonosan mozog CAD és térinformatikai programokban. Linuxszal történő ismerkedést RedHat 5.2-vel kezdte, sajnos jelenleg csak kedvtelésből foglalkozik vele.

Ideglelés

David azokra az olvasói levelekre reagál, amelyek legutóbbi, az érdekekről, a hiábavalóságról és a levélszemétről szóló felvetéseire érkeztek.

Amennyiben a befutó levelek alapján megítélhetem, a legutóbb elővett témákkal igencsak kényes helyre tenyereltem. Elsőként a terjesztésekben található könyvtárak változatainak kezelésével kapcsolatos gondokat említeném, illetve azt, hogy különféle rendszermagokat kell fenntartanom, például azért, hogy minden szükséges programot le tudjak fordítani, és azok fussanak is. A jelek szerint nem én vagyok az egyetlen. Talán ha kérvényeznénk a programozóktól, hogy ne mindig a legújabb-legszebb változatot használják, előbbre jutnánk – ebben a lapszámban pedig ez fog történni. Programozók, ha figyeltek, segítsetek nekünk, felhasználóknak!
A második dolog a levélszemét kérdése volt. Szomorúan láttuk, hogy az ORBS után hamarosan az ORBZ is eltűnik a süllyesztőben. Nyomukban azóta több feketelista is felbukkant, ám sikerük nyomkövetés nélkül kétséges. A Razor adatbázis is gyanús – több levelezési listáról érkező levelet is halálra ítélt, holott nem kéretlenül érkeztek. A különféle listák és adatbázisok tehát enynyit tudnak. Megismerkedtem ugyanakkor egy újabb ígéretes és jól testreszabható szemétszűrő csomaggal. Kiderül, hogy mi sült ki belőle – lásd lejjebb a SpamAssassinról szóló részt.
Végül sokan szeretnének búcsút inteni a Windowsnak, ám a Quicken csomag egyelőre pótolhatatlan. A pénzügyi csomagok lehangolók, érdektelenek, és kevés programozó vállalja azt az öngyilkossággal felérő elhatározást, hogy ilyesféle programot fejlesszen. Ám ettől még szükség van rájuk. Az egyik ígéretes csomag nyílt forrásúból kereskedelmibe ment át. Kereskedelmi csomagokról nem írok, de akinek személyes pénzügyi alkalmazásra van szüksége, az AppGen oldalán (➲ http://www.appgen.com) megismerkedhet a MoneyDance programmal.

dnotify

A segédprogram békésen szunyókál a háttérben, amíg az általa figyelt könyvtárak valamelyikében valaki előre meghatározott módon el nem ér egy fájlt vagy végre nem hajt valamilyen módosítást. Ekkor az előre megadott parancsot hajtja végre. Különösen behatolásérzékelő rendszerekben tehet rendkívül értékes szolgálatot. Valaki éppen a saját kis cuccait pakolgatja fel hozzád? A `dnotify` azonnal értesít, amikor a megfelelő könyvtárban található fájlokhoz fért hozzá. Futtatásához glibc szükséges.
➲ http://www.student.lu.se/~nbi98oli

Mail::SpamAssassin

Rengeteg levélszűrőt próbálgattam, de egyik sem tökéletes. Úgy tűnik, a Vipul-féle Razor adatbázis valakiknek szúrta a szemét, így helyette a SpamAssassinnal próbáltam zöldágra vergődni, amely több tekintetben is emlékeztetett a Razorra. Érdemes megemlíteni, hogy a SpamAssassin könnyedén átállítható. A kipróbálás ideje alatt több száz kéretlen levelet fogott meg. Mindössze egy olyan szemét volt, amelyet 4,6 ponttal átengedett, és egyetlen baráti levelet fogott meg, 5,5 ponttal. A SpamAssassin segítségével fehér- és feketelisták is létrehozhatók. Ha vannak olyan barátaid, akik például Costa Ricában élnek, és címük acr.co.cr végződésű, címüket a fehérlistára teheted, miközben a többi acr.co.cr szemétkirály küldeményei egyenesen a */dev/null*-ba kerülnek. Tetszik! Részletesen olvashatsz a programról a 64. oldalon. A futtatáshoz szükségesek: Perl, Net::DNS, Mail::Internet és Net::SMTP Perl-modulok, valamint a Procmail.
➲ http://www.spamassassin.org

Remote Accounts Handler

Bash-parancsfájl, ami valamivel túllép a `gpasman` szolgáltatásain. GPG-vel titkosított formátumban listát vezet a távoli számlákról és hozzáférésekről, bemeneti átadott értékként a megfelelő távoli hozzáféréssel meghívva pedig kezdeményezhetjük is a kapcsolódást a távoli géphez. Ekkor elindítja a megfelelő alkalmazást – SSH, Telnet, FTP, SFTP, HTTP, MySQL –, és csatlakozik a másik géphez. A futtatáshoz szükségesek: Bash, expect, dialog (elhagyható), valamint GPG.
➲ http://www.entropika.net/racs

Penetrator

Perl-alkalmazás, segítségével indexelhetjük az összes állományt, majd kereséseket végezhetünk szavak alapján, akárcsak a htDig vagy más keresőmotorok révén. A különbség annyi, hogy a helyi merevlemezen dolgozik, illetve mindenhol, ahol olvasási jogosultsággal rendelkezünk. Ha – hozzám hasonlóan – több évnyi munkád fekszik szöveges fájlokban, és szavakra szeretnél bennük keresni, próbáld ki a Penetratort. Az első futtatás ugyan eltart egy ideig, az újabb bejegyzések létrehozása viszont már gyorsan megy. A külön SQL-szolgáltatások előnyeit kihasználva SQL-lekérdezésekkel a Penetrator segítsége nélkül is turkálhatsz az adatbázisban. A futtatáshoz szükségesek: Perl, DB_File, Getopt::Long és DBI::Pg (elhagyható) Perl-modulok.
➲ http://www.triptico.com/software/penetrator.html

yesClock

Érdekes óraötlet. Nemcsak az időt mondja meg, hanem – ha a beállítások között megadtad, hogy merre laksz – viszonylagos nappali, illetve éjjeli elhelyezkedésedet is. Csak úgy, a poén kedvéért. Futtatásához JVM2 szükséges.
➲ http://www.germane-software.com/software/yesClock

Ennyit erre a hónapra!

David A. Bandel
(dbandel@pananix.com) jelenleg Panamában él, Linux- és Unix-tanácsadással foglalkozik. Társszerzője a Que Special Edition: Using Caldera OpenLinux című könyvnek.

A Zope és az adatbázisok

Reuven bemutatja, hogy egyszerű Zope-helyünket milyen könnyedén át lehet alakítani oly módon, hogy az adatokat relációs adatbázisból olvassa, és oda is írja vissza.

Szinte mindenki, aki komoly weblapot szeretne készíteni, lapját előbb vagy utóbb relációs adatbázishoz szeretné majd kapcsolni. Meglehet, a relációs adatbázis módszer immár harmincéves, de igen rugalmas, biztonságos és gyors. Az adatbázisok használata lehetővé teszi, hogy a webalkalmazásunkhoz szükséges adatokat anélkül tárolhassuk és kérhessük le, hogy saját megmaradó tárolóréteget (persistent storage layer) kellene létrehoznunk. Ennek eredménye kevesebb hiba, nagyobb gyorsaság és sokkal nagyobb biztonság.

A Zope nevű objektumközpontú alkalmazáskiszolgáló, amelyet az utóbbi néhány hónapban vizsgáltunk, egy ZODB nevű beépített objektum-adatbázissal rendelkezik. A ZODB egyszerre hatékony és könnyen használható; a Zope-ban minden, a DTML-dokumentumokat és könyvtárakat is ideértve, ZODB objektumokként tárolódik. A valóságban a ZODB ezeket az adatbázis-elképzeléseket tranzakciók formájában támogatja, ami azt jelenti, hogy fontos adatok tárolására is felhasználhatjuk, és eközben biztosak lehetünk benne, hogy adatainkat egy hosszú, összetett lekérdezés alatt sem fogja senki módosítani. Számos esetben azonban a tárolandó és lekérdezendő adatok kezelésére nem a ZODB a legjobb választás. Ennek sokszor egyszerűen az az oka, hogy az adat már létezik, és mi azt szeretnénk, hogy a Zope elérje. Az is megeshet, hogy állandó tárolóréteget akarunk készíteni, de azt szeretnénk, hogy az emberek a Zope-on kívülről is el tudják érni. Esetleg adataink egyszerűen jobban illeszkednek a relációs, mint az objektumközpontú adatbázismodellhez. Végül előfordulhat, hogy szervezetünk IT-részlege azt akarja, hogy minden adat relációs adatbázisban legyen tárolva.

E felsorolt okok és helyzetek miatt az alap-Zope-telepítés meghatározza a ZSQL tagfüggvény-objektumot. Ebben a hónapban a ZSQL tagfüggvényeit és a Zope általános relációsadatbázis-illesztési lehetőségeit fogjuk megvizsgálni. Mint látni fogjuk, egyszerű Zope-lapjainkat igen könnyű úgy átalakítani, hogy az adatokat relációs adatbázisból olvassa, és oda is írja vissza.

Adatbázis-kapcsolatok

Mielőtt elkezdhetnénk egy adatbázissal dolgozni, hozzá kell kapcsolódnunk. A Zope alatt ezt egy adatbázis-kapcsolati objektum létrehozásával tehetjük meg. A Zope-oldalak tetszőleges számú ilyen objektumot tartalmazhatnak, és ezek mindegyike képes SQL-lekérdezéseket küldeni az adatbázisnak.

A Zope alapesetben egyetlen típusú adatbázis-kapcsolatot ismer, amely az egyszerű ***Gadfly*** relációs adatbázissal való kapcsolatot teszi lehetővé. Csakhogy míg a Gadfly nagyon jól megfelel a Zope adatbázis-kapcsolatainak bemutatására, sebesség és képességek terén semmilyen más relációs adatbázissal nem versenyezhet. Azt javaslom, a Gadflyt egy az egyben hagyjuk is ki, és azt az adatbázis-csatolót telepítsük, amelyhez majd csatlakozni szeretnénk.

Mivel én az irodai adatbázis-kiszolgálón PostgreSQL-kiszolgálót futtatok, úgy döntöttem, hogy az Interneten jelenleg elérhető számos PostgreSQL-átalakító közül a `psycopg` adatbázis-átalakítót telepítem fel (a `psycopg`-ról további tájékozódási pontot a ***Kapcsolódó címek*** között találhatunk). Amikor ezt (vagy valamelyik másik) csomagot telepítjük, ne feledjük, hogy a Zope általában saját Python-változattal érkezik, ami a rendszerünkön esetleg fent lévő minden más másolattól független. Ez azt jelenti, hogy a `psycopg`-t a Zope-ban (a `$ZOPE/bin/python` által) megadott Python-könyvtárba kell telepítenünk, nem pedig a ***/usr/local/bin/python*** vagy a ***/usr/bin/python*** könyvtárak alá.

Mielőtt telepítenénk a `psycopg`-ot, telepítenünk kell az eGenix által írt és terjesztett `mxDateTime` osztályt. Ez a csomag lehetővé teszi, hogy a jelenlegi Unix-időkorlátokon (ami 1970-ben kezdődik és 2038-ban végződik) túli dátumokkal és időpontokkal is dolgozni tudjunk, illetve számos olyan kényelmes eljárást tesz elérhetővé, amelyek segítségével különféle dátum- és időformátumokkal dolgozhatunk. Ezt a modult akkor is fel kell telepítenünk, ha nem akarjuk használni, máskülönben a `psycopg` nem fog helyesen települni. A `mxDateTime` a ➲ http://www.egenix.com/files/python/eGenix-mx-Extensions.html címről tölthető le.

Figyeljünk rá, hogy nekünk a ***base*** (alap) csomagra van szükségünk (amely ingyenes), és nem az üzleti csomagra. Jobb, ha nem az `mxDateTime` RPM-változatát töltjük le, még akkor sem, ha RPM-alapú terjesztésünk van. Ennek az az oka, hogy a fordítást és a könyvtárak telepítését a Zope Python-fájában kell elvégeznünk, és nem a rendszer Python-fájában.

Az `mxBase` csomag letöltése és kicsomagolása után a telepítéshez az ***mxBase*** könyvtárba kell váltanunk, majd ki kell adnunk a következő parancsot:

```
$ZOPE/bin/python setup.py install
```

Az előbbi sor lefordítja az `mx`-modult, és Python-változatunkba telepíti.

A psycopg telepítése

Már közel járunk a `psycopg` telepítéséhez, amely Python- és C-kód keverékében íródott, és szüksége van a PostgreSQL fejlesztési (development) könyvtáraira. Ha a PostgreSQL-t RPM-ből telepítjük, szükségünk lesz az általunk használt PostgreSQL-változatnak megfelelő `postgresql-devel` RPM-csomagra is. A folyamat során az új fájlok általában a ***/usr/local/pgsql*** és a ***/usr/include/pgsql*** könyvtárakba kerülnek, bár néhány telepítés mindkét elérési útban `postgresql`-t használ `pgsql` helyett.

Most töltsük le a `psycopg` forráskódját a ➲ http://initd.org/pub/software/psycopg címről! Én a 1.0.4-es változatot töltöttem le, azonban úgy láttam, pár hetente kijön egy-egy új változat, ezért ha lehetséges, a legfrissebbet töltsük le. A `psycopg`

kicsomagolásához és telepítéséhez el kell készítenünk a `makesetup` parancsfájlt (ez jelenleg a legfrissebb Zope 2.5b1-ben a `$ZOPE/lib/python2.1/config` alá települ):

```
chmod 775 $ZOPE/lib/python2.1/config
```

A `psycopg` beállításához váltsunk a forráskönyvtárába, és gépeljük be a következőket:

```
./configure
  --with-python=$ZOPE/bin/python
  --with-zope=$ZOPE
  --with-mxdatetime-includes=$ZOPE/lib/python2.1/
    ➥site-packages/mx/DateTime/mxDateTime
  --with-postgres-includes=/usr/include/pgsql
```

Természetesen az elérési utakat a saját telepítésünknek megfelelően meg kell változtatnunk, különös figyelmet szentelve a Python-változatszámoknak (jelen esetben ez 2.1) és a PostgreSQL ***include*** könyvtárának.
Bár továbbra is meg vagyok győződve róla, hogy valamilyen `configure` kapcsolóval is meg lehetne oldani a dolgot, egyelőre úgy tűnik, a `Makefile`-t kézzel kell átszerkesztenünk, hogy az új header könyvtárat a `CFLAGS` változóhoz hozzáadhassuk. Kedvenc szövegszerkesztőnkkel nyissuk meg a `Makefile`-t, és a `CFLAGS` meghatározáshoz (az én változatomban a 90. sor) adjuk hozzá `$ZOPE/include/python2.1`-ben található headereket. Ha tehát a `$ZOPE` nálunk */**usr**/**local**/**zope***, a `CFLAGS`-hoz a következőket kell adnunk:

```
-I/usr/local/zope/include/python2.1
```

Mentsük a `Makefile`-t, majd telepítsük a `psycopg`-ot:

```
make && make install && make install-zope
```

A fentiek lefordítják a `psycopg`-ot, majd ***$ZOPE*** könyvtárunkba telepítik.
Végül a `psycopg` megosztott programkönyvtárát (***psycopgmodule.so***) a ***$ZOPE**/**lib**/**python2.1**/**site-packages*** könyvtárból helyezzük át a ***$ZOPE**/**lib**/**python2.1**/**lib-dynload***/-ba.

A psycopg beállítása

A Zope újraindítása után a `psycopg`-ot mindjárt ki is próbálhatjuk – egy új termék saját könyvtárba való telepítésével (sajnos a Zope újraindítása az egyetlen lehetőségünk, hogy a rendszerrel egy új termék telepítésének megtörténtét közöljük). A telepítendő termék neve ***Z Psycopg Database Connection***, és a Zope kezelőfelület jobb felső sarkában az ***Add product*** menüben találjuk.
Minden adatbáziskapcsolat-objektum egy távoli gép egyetlen adatbázishoz enged hozzáférést, egyetlen felhasználónévvel és jelszóval. Ez azt jelenti, hogy ha az adatot két különböző adatbázis között osztottuk meg (avagy különböző adatbázis-kiszolgálók között), akkor két kapcsolatobjektumra lesz szükségünk.
Amikor a ***Z Psycopg Database Connection***-t kiválasztjuk az ***Add product*** menüből, a rendszer feltesz néhány, adatbázis-kapcsolatunkra vonatkozó alapvető kérdést. Az ID-t (amelynek minden könyvtárban egyedinek kell lennie) és a címet (ez a kezelőfelületen jelenik majd meg), valamint az adatbázis-kapcsolat karaktersorozatát be kell írnunk. Ez a kapcsolatkaraktersorozat adja meg a Zope-nak, hogyan tud kapcsolatba lépni a PostgreSQL-kiszolgálóval. Irodámban az atf adatbázis a `databases`-en található PostgreSQL-kiszolgálón helyezkedik el, ahová `reuven` néven, jelszó nélkül kapcsolódhatok. Ennek megfelelően a következő kapcsolódási karaktersorozatot használom:

```
host=databases dbname=atf user=reuven
```

Ha kívánjuk, a maradék részeket meghagyhatjuk az alapértelmezett értéken. Kattintsunk az ***Add*** gombra, ezáltal visszatérünk abba a könyvtárba, ahová az új kapcsolatobjektumunkat illesztettük be.
A kapcsolatobjektumra kattintva néhány Zope-tábla jelenik meg, segítségükkel elvégezhetjük az adatbázis karbantartását. Ezek közül a négy legérdekesebb:

- ***Status***: ebből a táblából megtudhatjuk, hogy az adatbázis-kapcsolat nyitott-e (vagyis éppen kapcsolódik-e a PostgreSQL-kiszolgálóhoz vagy sem). Itt – ha szükséges – lehetőségünk nyílik ezt a kapcsolatot lezárni.
- ***Properties***: megváltoztathatjuk azokat a beállításokat, amelyeket az adatbáziskapcsolat-objektum eredeti létrehozásakor írtunk be. Különösen hasznos lehet ez a rész, ha az adatbázist másik kiszolgálóra visszük át, vagy megváltoztatjuk az eléréshez szükséges jelszót.
- ***Test***: az adatbázis-kapcsolatot próbálhatjuk ki azáltal, hogy tetszőleges SQL-lekérdezést küldünk ki rá. Természetesen a lekérdezésnek érvényesnek kell lennie; ha hibás SQL-lekérdezést küldünk ki, vagy olyan táblát címzünk meg, amely nem létezik, a PostgreSQL-kiszolgáló által visszaadott megfelelő hibajelzést kapjuk vissza. Például begépelhetjük a `SELECT * FROM pg_database;` parancsot. Bármilyen SQL-utasítást bevihetünk ezen a dobozon keresztül, ami jól jöhet az adatbázis kipróbálásakor, főleg, ha nem rendelkezünk közvetlen Telnet- vagy SSH-kapcsolattal. Amikor `INSERT` vagy `UPDATE` lekérdezést gépelünk be, a Zope jelzi, hogy a lekérdezés nem ad vissza eredményt. Mint általában, most sem érdemes `SELECT *` formájú lekérdezéseket írni, legfeljebb egyértelmű esetben, nehogy végül meglepődjünk az eredményoszlopok sorrendje vagy neve miatt.
- ***Browse***: a böngésző (browse) táblán a PostgreSQL adatbázistábláit nézegethetjük végig, amely Zope-fastílusban listázza a táblákat és azok mezőit.

ZSQL tagfüggvények

Most, hogy az adatbázis kapcsolat már él, egy vagy több ZSQL tagfüggvényt is készíthetünk. Minden ZSQL tagfüggvény egyetlen SQL-lekérdezés (ha kívánjuk, változó számú értékkel), amely az adott kapcsolattal dolgozik.
Készítsünk egy ZSQL tagfüggvényt, amellyel új nevet adhatunk a telefonkönyvhöz. Természetesen ehhez az adatbázisban előbb meg kell határoznunk a megfelelő táblát. A táblát könnyen elkészíthetjük, ha az *1. lista* tartalmát elküldjük a PostgreSQL-nek – akár az adatbázis kapcsolatunk próbatábláján, akár a hagyományos `psql` parancssoros felületen.
Ha olyankor akarunk ZSQL tagfüggvényeket beilleszteni, amikor nincs elérhető adatbázis-kapcsolat, a Zope hibaüzenetet fog küldeni, jelezvén, hogy egyetlen megfelelő adatbázis-kapcsolatot sem talált.
A Zope beépített „szerzeményezés” rendszerének megfelelően a ZSQL tagfüggvényei bármely adatbázis-kapcsolatot felhasználhatják, amely a Zope rendszerében felettük áll. A felhasz-

1. lista A tábla létrehozása

```
CREATE TABLE AddressBook (
  person_id         SERIAL,
  first_name        TEXT    NOT NULL,
  last_name         TEXT    NOT NULL,
  address1          TEXT    NOT NULL,
  address2          TEXT    NULL,
  ➥- Nem mindenkinek kell a mÆsodik sor
  city              TEXT    NOT NULL,
  state_province    TEXT    NULL,
  ➥- Nem mindenki Øl az USA-ban
  postal_code       TEXT    NULL,
  ➥- Nem minden orszÆgban van
  phone_number      TEXT    NOT NULL,
  ➥- Nem mindenki Øl az USA-ban
  fax_number        TEXT    NULL,
  ➥- Nem mindenkinek van faxgØpe
  cell_number       TEXT    NULL,
  ➥- Nem mindenkinek van mobiltelefonja
  PRIMARY KEY(person_id)
);
```

náló így minden tagfüggvényhez különféle adatbázisokat rendelhet – ez a különböző adatbázisokban fellelhető adat egyetlen alkalmazásba egyesítését, vagy a teljes weblap átültetését egy másik adatbázistípusra teszi lehetővé.
A ZSQL tagfüggvények létrehozásához váltsunk abba könyvtárba, ahol adatbázis-kapcsolatunkat létrehoztuk, és válasszuk az ***Add product*** menü ***ZSQL method*** pontját. Ismét meg kell adnunk pár adatot: az ID-t (amely az objektum könyvtárán belüli egyedi azonosítását szolgálja), a címet (ez a kezelőfelületen fog látszani), a kapcsolókat (melyeket majd a következő részben tárgyaljuk), végül magát az SQL-t. Az SQL-lekérdezés egyszerű vagy tetszés szerinti bonyolultságú lehet, és `INSERT`, `UPDATE` vagy `DELETE` utasítást is végrehajthat.
Miután ZSQL tagfüggvényünket a rendszerhez adtuk, rákattintva számos Zope-tábla kerül elő. Az egyik ezek közül `test` címkével rendelkezik és – miként azt sejteni lehet – a lekérdezés kipróbálására szolgál. Ha lekérdezésünk értékeket is vár, egy HTML-űrlapon megadhatjuk őket. Ha nem, akkor egyszerűen csak a ***Submit Query*** gombra kell böknünk. Ezután, akárcsak adatbáziskapcsolat-objektumunk próbatábláján, egy HTML-formátumú táblát kapunk vissza, amely lekérdezésünk eredményeit hivatott megjeleníteni, vagy jelzi, hogy lekérdezésünk nem adott vissza eredményt.
Minden egyes kiadandó lekérdezéshez elkészíthetjük a megfelelő ZSQL tagfüggvényt. Bár ez elsőre kicsit nehézkesnek tűnhet, valójában igen rugalmas és elegáns megoldás, amelyet egyre inkább megtanulok értékelni. Ha körülbelül húsz lekérdezés kiadására számítok, egy webalkalmazásban mindegyiket külön ZSQL tagfüggvénybe helyezem, majd a DTML-lapokról ezeket a tagfüggvényeket hívom meg.
A DTML-lapokon a ZSQL tagfüggvények eredményét `<dtml-in>` tag formában kaphatjuk meg. Ha például a következő lekérdezést megvalósító ZSQL tagfüggvényt szeretném használni:

```
SELECT first_name, last_name, phone_number,
       fax_number, cell_number
  FROM AddressBook
 ORDER BY last_name, first_name
```

Ha ennek a ZSQL-nek a ***names-and-phone-numbers*** nevet adom, a 2. listában (36. CD Magazin/Zope könyvtár) látható kóddal hívhatom meg egy DTML-dokumentumból. Mindössze néhány soros DTML-kóddal képesek voltunk létrehozni egy egyszerű (de hasznos és rugalmas) ZSQL tagfüggvényt. Nézzük, hogyan is működik!
Amikor a Zope ehhez a DTML-dokumentumhoz tartozó kérelmet kap, értelmezi a DTML-t és a benne található összes tagot végrehajtja. A `<dtml-in>` ciklusszerkezet egy adatsort vár bemenetként – jelen esetben ez az adatsor a ***names-and-phone-numbers*** tagfüggvény meghívásából kapott eredmény. A `<dtml-in>` tag a visszaadott halmaz minden egyes oszlopához egyúttal egy-egy változót is rendel. Ez az oka annak, hogy a `<dtml-var first_name>` tagot később a felhasználó első nevének kiírásához fel tudjuk használni; a Zope a `first_name` oszlop értékét önműködően a `first_name` nevű változóhoz rendeli.
Hogy elkerülhessük a felesleges és az üres sorok kiírását, a `<dtml-if>` tagot használjuk annak eldöntésére, hogy a PostgreSQL értékes, vagy NULL, azaz üres értéket adott-e vissza.

A ZSQL értékei

Az már nyilvánvaló, hogyan használhatjuk a ZSQL tagfüggvényeket és a DTML-t, ha mindig ugyanazt a lekérdezést akarjuk kiadni. Ha azonban az alap-lekérdezésünket minden egyes futás során módosítani szeretnénk, egy vagy több értéket is meg kell adnunk.
Ha valakiről vezetékneve (vagy vezetéknevének egy része alapján) szeretnénk adatott lekérni, az SQL szabványos kifejezéseit kihasználva a következő példához hasonló ZSQL tagfüggvényt kellene megadnunk:

```
SELECT first_name, last_name, phone_number,
       fax_number, cell_number
  FROM AddressBook
 WHERE last_name LIKE XXXXXX
ORDER BY last_name, first_name
```

A DTML-ben az XXXXXX helyére a `<dtml-sqlvar>` tag kerül, amely a helyes idézőjelezést is önműködően elvégzi helyettünk. A felhasznált SQL-változót és annak típusát is meg kell adnunk:

```
SELECT first_name, last_name, phone_number,
       fax_number, cell_number
  FROM AddressBook
  WHERE last_name LIKE <dtml-sqlvar
  name_sqlregexp type="string">
ORDER BY last_name, first_name
```

Ahhoz, hogy a fenti ZSQL-példa helyesen működjön, a tagfüggvény létrehozásakor az értéket (`name_sqlregexp`) a megfelelő szövegmezőben meg kell neveznünk. A Zope ennek a változónak veszi majd az értékét, lekérdezésünkbe helyezi, majd az eredményeket lekéri.
Még többet is kihozhatunk a Zope-ból, ha a `<dtml-sqltest>` tagot használjuk, ami a `<dtml-sqlvar>` tagjához hasonló módon működik:

```
SELECT first_name, last_name, phone_number,
       fax_number, cell_number
```

```
    FROM AddressBook
   WHERE <dtml-sqltest name_sqlregexp
         op="like" type="string">
ORDER BY last_name, first_name
```

Ha a fenti lekérdezést a `select_by_last_name` nevű ZSQL tagfüggvényben tároljuk, akkor a Zope önműködően el tudja készíteni a DTML-dokumentum vázát, azaz egy olyan dokumentumot, amelyben a felhasználók keresési feltételeket adhatnak meg, és az eredményt is megnézhetik. Ehhez egyszerűen csak ki kell választani a ***Z Search Interface*** terméket az ***Add product*** listából. Itt a rendszeren található összes kereshető objektumból választani tudunk, ideértve az éppen most elkészült ZSQL tagfüggvényt (`select_by_last_name`). Válasszuk ki, és adjunk neki egy ID-t (én a `search_by_last_name` nevet használtam). Névként az `input ID`-t adtam, ami elég beszédes egy olyan HTML-űrlap esetében, amely bemenetként szolgál a `search_by_last_name`-hez (ezt pedig `search_by_last_name_form`-nak neveztem el). A Zope korszerű változataiban azt is megadhatjuk, hogy a rendszer DTML tagfüggvényeket vagy lapsablonokat (page template) készítsen – mi most az előbbit szeretnénk.
Az ***Add*** gombra kattintva a pillanatnyi könyvtárban két új DTML tagfüggvény jön létre a korábban megadott neveknek megfelelően. Az `input ID` címre kattintva egy egyszerű HTML-űrlapot jelenít meg, melybe SQL-szabványos kifejezést gépelhetünk. A ***Submit*** gombra mutatva a lekérdezés elküldődik a `search_by_last_name` DTML tagfüggvényhez, amely viszont a mi ZSQL tagfüggvényünket hívja meg (`select_by_last_name`), így a lekérdezés végül a PostgreSQL-hez kerül. A PostgreSQL az eredményeket visszaadja a `select_by_last_name`-nek, amely az eredményhalmazt a `search_by_last_name`-hez továbbítja, ami ezután böngészőnkön megjeleníti őket.
Természetesen a létrehozott DTML tagfüggvényeket módosíthatjuk is, hogy jobban tükrözzék a lapunk stílusát. A Zope által önműködően készített DTML-lapokat át is másolhatjuk, példaként használva őket saját adatbázis-lekérdezéseinkhez.

Beszúrás

Az egyetlen nagyobb feladat, ami még hátravan, az `INPUT` lekérdezés megvalósítása, amely elemeket ad az adatbázishoz. Szerencsére a megoldás meglehetősen egyszerű: készítünk egy ZSQL tagfüggvényt, amely sorokat szúr az adatbázisba. Ezután egy DTML-dokumentumot kell megalkotnunk, amely a HTML-űrlap elemeit egy másik DTML-dokumentumnak adja át. Ez a második dokumentum meghívja a ZSQL tagfüggvényünkhöz tartozó `<dtml-call>`-t. Voilà, rekordunk máris bekerült az adatbázisba.
A 3. lista (36. CD Magazin/Zope könyvtár) mutatja be a létrehozandó ZSQL tagfüggvényt, amelyeket `insert_address_data` névre kereszteltem. Ezután egy egyszerű DTML-dokumentumot hozunk létre, amely egyetlen HTML-űrlapot tartalmaz (lásd a 4. listát, 36. CD Magazin/Zope könyvtár).
Végül elkészítjük az `insert_address` DTML-dokumentumot, amely az `insert_address_form` adatait fogadja, és az adatokat a `insert_address_data` ZSQL tagfüggvényhez továbbítja:

```
<dtml-var standard_html_header>
<h2><dtml-var title_or_id></h2>

<dtml-try>
    <dtml-call insert_address_data>
<dtml-except>
    <p>Sorry, but the INSERT didn·t work.</p>
<dtml-else>
    <p>Successfully inserted!
</dtml-try>

<dtml-var standard_html_footer>
```

A felhasználók ettől kezdve az `insert_address_form` által megadott HTML-űrlap felhasználásával képesek lesznek adatokat bevinni a PostgreSQL-táblába, és a `search_by_last_name_form` segítségével lehetőségük nyílik lekérdezni az adatokat. Elég lenyűgöző, hogy ilyen kevés fájllal ilyen sokat meg tudtunk csinálni, ráadásul ahhoz, hogy mindez működjön, még szövegszerkesztőre sem volt szükségük, mindössze a webböngészőnket kellett használnunk.

Összegzés

Bár nem tökéletesek, a ZSQL tagfüggvényeket igen elegáns módszernek tartom a HTML-lapok és a mögöttük elhelyezkedő adatbázis összekapcsolására. A ZSQL egy újabb példája annak, hogy a Zope milyen rugalmas, finom megközelítése a webfejlesztésnek, bár az is igaz, okoz némi fejtörést, mire mindez tisztává és világossá válik. Ha valaki már eleve ismeri a DTML-t és az SQL-t, annak nem fog nehézséget okozni Zope-alkalmazásaiba ZSQL tagfüggvény segítségével adatbázisokat beépíteni, sőt, az is megoldható, hogy a munkát egy honlapon belül megosszuk az SQL-t ismerő (és ZSQL tagfüggvényekkel dolgozó), illetve az őket meghívó DTML-tagfüggvényekkel foglalkozó munkatársak között.

Linux Journal 2002. május, 97. szám

Reuven M. Lerner (reuven@lerner.co.il) kisebb, webes és internetes módszerekkel foglalkozó tanácsadó cég tulajdonosa és vezetője. A cikk megjelenésének időpontjában valószínűleg már végleg elkészült Core Perl című könyvével, melyet idén jelentet meg a Prentice-Hall. Az ATF honlapon érhető el (➲ http://www.lerner.co.il/atf/).

Kapcsolódó címek

A Zope honlapja ➲ http://www.zope.org
Nagyszámú leírást rejt, többek között általános adatbázis- és ZSQL-felhasználási útmutatót. A Python nyelv, melyben a Zope nagy része íródott, a ➲ http://www.python.org címen érhető el.

A Pythonhoz és Zope-hoz írt psycopg-csatoló a ➲ http://initd.org címen érhető el. Az írásunkban olvasható telepítési utasítások nagyrészt a ➲ http://initd.org/pub/software/psycopg/FAQ megjegyzésein alapulnak. A PostgreSQL honlapja a ➲ http://www.postgresql.org címen található.

Adjunk életet programozással!

Íme néhány megunhatatlan játék a programozással most ismerkedők számára.

A programozók olyanok, mint az istenek, François. Kezdetben semmijük sincs, csak egy gondolat a fejükben, átírnak néhány bitet vagy bájtot, és voilà! A káoszból az élet egy új formáját hozzák létre a Linux-rendszeren. Ez valóban bámulatos, mon ami. Igen, természetesen igazad van, François. Ez kissé félelmetes is. A hatalom felelősség! Ám létrehozni valamit a semmiből nemcsak félelmetes, de mámorító érzés is. A mámorról jut eszembe, hogy mindjárt megérkeznek a vendégeink. A toszkánai megfelelő lesz mára. Még mindig nem indultál el a pincébe az 1997-es Vino Nobile di Montepulcianoért? Siess, François, rögtön betoppannak!
Már itt is vannak! Bon soir, mes amis! François éppen a pincében van, hogy felhozza a mai alkalomra szánt bort. Mielőtt megérkeztetek, éppen a digitális teremtés örömeiről, a programozásról társalogtunk, és ez egyben mai közös beszélgetésünk témája.
A Linux, amelyet az étteremben és otthonainkban használunk, azoknak a tehetséges és lelkes programozóknak az alkotásaiból áll, akik úgy határoztak, hogy munkájukkal hozzájárulnak ennek a közösségnek az építéséhez. Alkotásaik néha a kibernetikai lét bizonyos fajtáinak – robotoknak és egyéb mesterséges lényeknek – adnak életet. Ha bármikor felmerült bennetek a gondolat, hogy csatlakoznátok ennek az alkotócsoportnak a munkájához, a mai menü nektek szól. Á, François, merci, tölts, légy szíves, a vendégeinknek! Ne sajnáld tőlük!
A játékok képezik az új dolgok megtanulásának egyik módszerét, ennek eszköze pedig a Java lehet. Az IBM Robocode nevű programozási környezetét kínálja, amellyel képernyőnk küzdőterén csatázó Java-robotokat hozhatunk létre. Minden csata több fordulóból áll, és az eredményről kimutatás is készül, hogy kedvenc robotunk pillanatnyi állását nyomon követhessük. Ez az adat segít abban, hogy a rengeteg állítható kapcsoló variálásával még hatékonyabbá tegyük a robotot. Vajon, ha a robot beállított száz lépés helyett minden harmincadik lépésnél megáll és megfordul, ez javítja a siker esélyét? Kiderül, ha beállítjuk a kapcsolót, lefordítjuk, és kipróbáljuk a programot. Remek kezdési lehetőség azok számára, akik most ismerkednek a Java nyelvvel – és nem könnyű elszakadni tőle.
A Robocode-dal való munka elkezdéséhez először a Java fejlesztői készletre lesz szükségünk. Ezt ingyen beszerezhetjük a ➲ http://java.sun.com/j2se címről, csak a megfelelő csomagot kell letöltenünk. Én rendszeremen az RPM-csomagot használtam, de más Linux-rendszerekhez GNU `tar` csomag is rendelkezésre áll. A telepítéssel kapcsolatban talán érdemes figyelmeztetnem titeket arra, hogy a bináris állományok nem a szabványos ***bin***-könyvtárakba kerülnek.
A folytatáshoz a `PATH` változót az alábbi módon meg kellett változtatnom:

```
export PATH=$PATH:/usr/java
➥/jdk.1.3.1_02/jre/bin
```

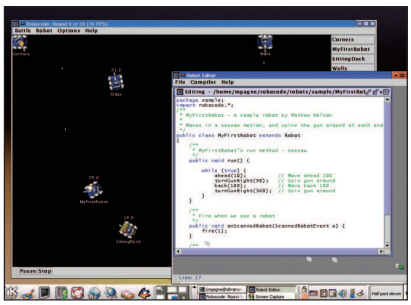
Ha a Java telepítésén sikeresen túljutottunk, az IBM alphaworks honlapjáról le kell töltenünk a Robocode csomagját, amelyet a ➲ http://robocode.alphaworks.ibm.com/home/home.html címen érhetünk el.
A telepítés folyamatát fájdalommentesnek fogjuk találni, mivel programfordításra egyáltalán nincs szükség. Egyszerűen a Java segítségével a telepítőprogramot az alábbi paranccsal kell futtatni:

```
java -jar robocode-setup.jar
```

Az első dolog, amibe beleütközünk, egy ablak, ami a felhasználási szerződés elfogadását kéri. A telepítési könyvtárra vonatkozó javaslat követi – én elfogadtam az alapértelmezett értéket, ami a saját könyvtáramban létrehozott ***robocode*** nevű alkönyvtár. A telepítés befejezésével a program a következő paranccsal indítható:

```
cd $HOME/robocode ./robocode.sh
```

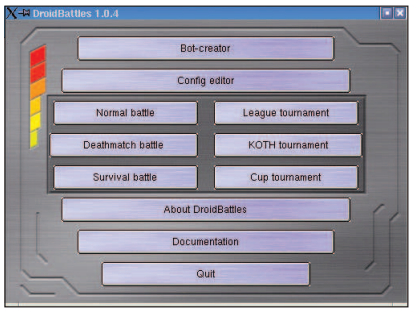
A program elindítása után a Robocode harcmező jelenik meg, itt vívják csatáikat a robottankok. Válasszuk ki a ***Battle*** (csata) menü ***New*** (új) menüpontját. Megjelenik egy oldal, amelyen a következő csatában résztvevő robotokat választhatjuk ki. Mindegyik neve előtt a „sample” (minta) előtag szerepel, és bizonyos előre beállított tulajdonsághalmazzal bírnak. Kezdésnek válasszuk például a ***sample.MyFirstRobot*** és a



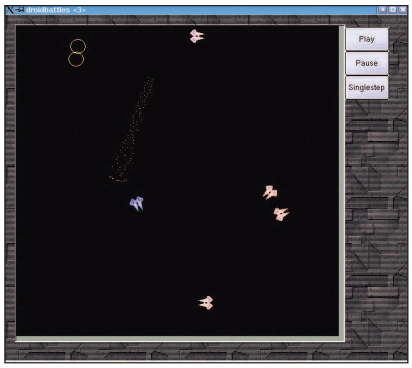
1. kép Robotok szerkesztése Robocode-ban

sample.Spinbot nevű robotokat. Mielőtt a ***Start battle*** (a csata kezdése) gomb megnyomásával továbbmennénk, válasszuk ki a ***Battlefield*** (harcmező) fület. Saját 1024×768-as asztalomon egy szerény 800×600-as csatateret választottam. Végül a ***Rules*** (szabályok) fülön kell a követendő szabályokat beállítanunk, például a fegyver hűtési gyorsaságát és a tétlenség idejét. Ezeknél járjunk el körültekintően. Kezdetnek elfogadhatjuk az alapbeállított értékeket is. Készen állunk? Allons, mes amis! Nyomjuk meg a ***Start battle*** gombot.
Kaphatunk olyan robotot, mint a Sitting Duck (ücsörgő kacsa), amelyik csak ül és várja, hogy a csata szele elsodorja. Választhatjuk a Crazy (bolondos) nevűt is, amely látszólag véletlenszerűen mozog, és nagyrészt rendszertelenül adja le a lövéseit. Mindegyik betölthető, és viselkedésüket az eredetileg beprogramozott beállításoktól különböző tulajdonságok megadásával módosíthatjuk. Amennyiben – hozzám hasonlóan – te sem vagy megelégedve a csata eredményével, itt az ideje egy kis kódolásnak és robotunk módosításának. Az első alkalommal, amikor úgy döntünk, hogy robotunkat módosítjuk, a Robocode üzemképessé teszi Jikes-ot, a fordítóprogramot (és elküld minket, hogy igyunk egy csésze kávét). Oui, mes amis, én is éppen ezt a kérdést tettem fel magamnak. Milyen idősnek kell lennie egy programozónak, míg az igazi Javával próbálkozhat? Az *első képen* a robotszerkesztő és egy folyamatban lévő játék egy-egy képét láthatjuk. A tanulás legegyszerűbb módját a példarobotokkal való kísérletezgetés jelenti, ha ebben

már otthonosak vagyunk, saját példányunk elkészítését is megkockáztathatjuk. A fordítóprogram üzembe helyezésével a Robocode teljes fejlesztői környezete is rendelkezésünkre áll. Kattintsunk a ***File*** (fájl) menü ***Open*** (megnyitás) menüpontjára, és válasszuk ki a „sample" fájlt. Egy új listát kapunk, amelyben az összes mintarobot (mint például a



2. kép A DroidBattle irányítópultja



3. kép A DroidBattle harcmezeje

Ramfire) szerepel. Minden robothoz tartozó kód meglehetősen jó leírással bír, így működési elvét még egy kezdő programozó is hamar megértheti. Például mit tegyen a robotunk akkor, ha falnak ütközik:

```
/**
       * onHitWall:  A falnak
➥ tk zØs kezelØse.
       */
      public void
onHitWall(HitWallEvent e)
      {
            // PattanÆs!

reverseDirection();
      }
```

A robotok osztályának különböző eljárásai a `snap to`, `catch on to`, `ahead()`, `back()`, `fire()`, `fireBullet()`, `donothing()` és így tovább. Az API leírását a telepítési könyvtárban lévő ***javadoc/index.html*** néven találhatjuk meg. François, ha lennél kedves vendégeink poharait újratölteni… Szeretnék veletek megosztani még egy hasonló fejlesztői környezetet.

Andreas Agorander DroidBattles programja a Robocode-éhoz hasonló alapelvre épül, bár programozási nyelvként nem Javát, hanem egy assemblyhez hasonló nyelvet használ. Ezen az alacsony szinten olyan közel kerülünk programunk lelkéhez, amennyire csak lehetséges. A program maga a 2.X Qt programozói könyvtárakra épül, így azoknak, akik a KDE 2.X-változatát használják, nem lehet gondjuk a program lefordításával. Mielőtt még azonban, mes amis, hozzákezdenétek a fordításhoz, először el kell jutnia hozzátok, illetve nektek kell eljutnotok hozzá.

A legfrissebb forráskód a ➲ http://www.bluefire.nu/droidbattles/index.html oldalon érhető el.

A telepítés közbeni egyetlen furcsaság az volt, hogy az archívumban lévő állományok dátuma jövőbeli értékre lett állítva, ami némi bánatot okozott nekem, amikor eljutottam a `./configure` lépéshez.

```
tar -xzvf droidbattles-
➥1.0.4.tar.gz
cd droidbattles-1.0.4
./configure
make
make install
```

Ezt a programot az teszi elragadóvá, hogy a robotot a kódban is ugyanúgy építhetjük fel, mint a valóságban: processzorokat, memóriát, motorokat, plazmafegyvereket és optikai letapogatókat adhatunk hozzá. Amikor a `droidbattles &` parancs beírásával elindítjuk a programot, a *2. képen* láthatóhoz hasonló irányítópultot kapunk. A csatához legalább egy robotot létre kell hoznunk és be kell programoznunk. Az ablak felső részén található ***Bot-creator*** (robot létrehozása) gomb megnyomásával tehetjük meg az első lépést gépkatonánk megalkotása felé.

Bevásárlólistához hasonló felsorolást kapunk robotunk fizikai tulajdonságairól, valamint egy ablakot a kód számára. Azok számára, akik korábban még soha nem láttak assemblert, a forráskód egy kicsit furcsának tűnik majd, de az érzés hamar el fog múlni. Ezeket az utasításokat közvetlenül a processzor kapja – természetesen nem a gépünkben lévő, hanem a robotunkban elhelyezett. Az utasítások egyszerűek, ráadásul a DroidBattles-szel együtt érkező teljes leírás is a rendelkezésünkre áll, egyszerűen csak a ***Documentation*** gombra kell kattintanunk. Mellesleg ha továbbhaladva egyre több alkatrészt adunk a robothoz, a rendszer hagyja, hogy lefordítsuk, futtatni azonban már nem engedi. Vajon miért? – merül fel a jogos kérdés. A beállításszerkesztő adja meg azokat a kapcsolókat, amelyekhez tartanunk kell magunkat, ha robotot hozunk létre. Például megadjuk az eszközök vagy a memória legnagyobb mennyiségét, valamint ezeknek az egységeknek az árát. Továbbá egy robot csak meghatározott pénzmennyiségbe kerülhet, és minden egyes eszköz pénzbe kerül (ne aggódjatok, mes amis, nem valódi pénzről van szó). Az árat az adott eszköz hatékonysági foka határozza meg. A plazmafegyverek ára 50 és 3500 dollár között mozoghat. ezek az értékek teszik lehetővé, hogy a különböző jellegzetességeket meghatározott keretek között variálhassuk. Vajon javítja-e az esélyeinket egy csatában, robotunk a kétszer annyi plazmafegyverrel, de csak feleakkora számítási teljesítménnyel rendelkezik? Állítsuk be, mentsük a robot változtatásait, és próbáljuk ki!

Végül, ha mindezzel készen vagyunk, kattintsunk a ***Tests*** (tesztek) menüre a Bot-creatorban, és válasszuk a ***Quick battle*** menüpontot. Legalább két robotot kell a csatához kiválasztanunk, de akár ugyanahhoz a fajtához is tartozhatnak. A csatamező megjelenik (lásd a *3. képet*), és már indul is a játék. Kattintsunk a ***Play***-re (játék), és figyeljük a mókát!

A csata folyamata közben a ***Pause*** (szünet) gombra kattintva lépésenkénti üzemmódra kapcsolhatunk át. Próbaüzemmódban megjelenik egy hibakereső ablak, amelyben a memória lefoglalásáról, a regiszterek tartalmáról és az éppen futó utasításról kaphatunk tájékoztatást. Mindez rengeteg érdekességet és mulatságot rejt magában. Andreas honlapját is megtekinthetjük, amint épp egy DroidBattle-kiszolgálót helyez üzembe. Nos, mes amis, attól tartok, ma nagyon gyorsan letelt az időnk. Az egyik legnehezebb kérdés, amellyel szembesülnünk kell, amikor a programozáshoz hasonló útnak vágnunk neki, hogy mit is szeretnénk létrehozni. A Robocode és a DroidBattles két olyan program, amely keretet ad ehhez az elinduláshoz, és egy olyan küzdelemben enged részt venni, amelyben a kódolás inkább játéknak, semmint munkának tűnik.

Bon appétit!

Linux Journal 2002. június, 98. szám

Marcel Gagné (mggagne@salmar.com) Mississaugaban (Ontario, Kanada) él, a Salmar Consulting Inc. cég elnöke.