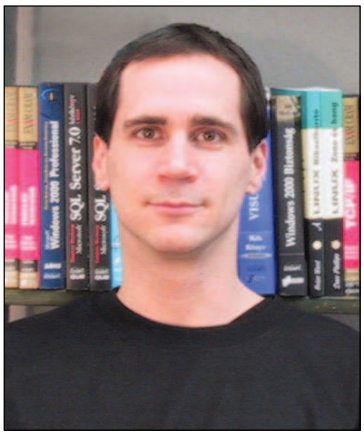
Örök vita, hogy melyik szót kell magyarítani, és melyiket kell jövevényszóként megtartani. Most nem szeretném újra elővenni a többször lerágott csontot, hogy miért is jobb hosszú távon szerkezetről és nem architektúráról beszélni, vagy hogy az informatikát tanulók miért értik meg könnyebben a gyorstár működésének alapjait, mint a cache szerkezetét. Ehelyett inkább egy részt szeretnék idézni a Magyar Helyesírási Szótárból:
„202. Az egymással érintkező nyelvek mindig adnak át egymásnak, és vesznek át egymástól közszavakat is, tulajdonneveket is.
a) Az idegen eredetű közszavak idő múltával jövevényszavakká válhatnak, azaz olyannyira meghonosodhatnak, hogy más nyelvből való származásuk feledésbe merül. A jövevényszavakat ezért mindig magyarosan írjuk.
Az olyan szavakat, amelyeknek más nyelvi eredete még többé-kevésbé nyilvánvaló, idegen szavakként tartjuk számon. Ha egy latin betűs írású nyelvből átvett, általános fogalmat jelölő idegen szó közkeletűvé válik, eredeti írásmódját a magyar kiejtést tükröző formával váltjuk fel. Az idegen szavakat tehát aszerint írjuk már magyar vagy még idegen írásmód szerint, hogy mennyire haladtak előre a jövevényszóvá válás útján...”
Érthető, nem? Tehát a jövevényszóvá válás feltétele, hogy olyannyira honosodjon meg az adott szó, hogy már ne is emlékezzünk rá, melyik nyelvből származik. Nyilván a szabályzat létrehozása óta a világ sokat változott, hiszen ennek értelmében semelyik számítástechnikai szót nem írhatnánk le magyar kiejtés szerint. Vagy mutassatok nekem egyetlen informatikust, aki nem tudja, hogy a fájl szó honnan ered. A szöveg szerint nyilvánvaló az ebből származó következtetés:
„A jövevényszavakat ezért mindig magyarosan írjuk.” Hogy mi jövevényszó, és mit kell még idegen eredetű szónak tekinteni, nos, erre nem kapunk útmutatást.
A további szöveg alapján azonban mondhatjuk, hogyha egy szó közkeletűvé válik, magyar kiejtés szerint kell írni. Közkeletű, de milyen körökben? Mert informatikusok között közkeletű a bútsztrep, a risztart, a félőr (aranyosak így leírva, nem?) és még rengeteg olyan szó, amit az informatikával nem, vagy csak érintőlegesen foglalkozók egyáltalán nem használnak.
Nyilván meg kell húzni a határt valahol, de félő, hogy akárhol húzzuk meg, erőszakot teszünk vagy a szaknyelven, vagy a magyar nyelven. A minap találkoztam egy hazai kiadó új könyvében egy féloldalas magyarázattal, amelyben kifejti, hogy miért is kell ma már a hacker helyett hekkert használni. Merthogy már általánosan használt szó.
Az már csak egy fura fintort csal az arcomra, hogy a következő sorban kijelenti, hogy a cracker szót viszont így, az angol írásmód szerint használja, nem pedig (a „testvérével” párhuzamosan) krekkernek.
Ebből is látszik, hogy az idegen eredetű és jövevényszavak körül nagyobb a kavarodás, mint az kívánatos volna, és ahogy egy ismerősöm mondta, arra aztán hiába várunk, hogy az akadémikusok teremtsenek rendet, hiszen egy-egy ilyen szó átrágása tovább tart, mint ameddig egyáltalán a szó eredete ismert. Azt pedig már tudjuk, hogy egy ilyen hosszú ideig nyújtott vita mit eredményez. A nyelvben gyökeret ver a hekkerborzalom, mint az öreg tölgy, utána pedig sopánkodhatunk, hogy „szép magyar anyanyelvünk”, meg hogy „egységes, egészséges nyelvtan”, ősi, szent nyelvünkre keresztet vethetünk.
Elismerem, azoknak, akik routerként tanulták az útválasztót, nehéz megszokni, hogy létezik ennek a fura szerkezetnek magyar neve is; más pedig büszkén vallja, hogy ő bizony a világ nagy hackerei közé tartozik, szemében sejtelmes csillogással és egy kis bizsergéssel a lapockái között, hiszen érzi a szemben álló fél csodálkozó félelmét az ismeretlen, de felettébb vonzó titulus kapcsán – annak a „hackernek” hiába magyarázzuk, hogy sok magyar párja van már ennek a szónak is, egyenesen sértőnek, trónfosztónak érzi, ha egyszerűen leszámítógépbetyározzuk vagy leszakizzuk. Igaz, ebben az esetben nem keverné össze minden riporter a szakembereket az önjelölt betörőkkel, sőt nem adna ki egy egyébként biztonsággal foglalkozó óriáscég olyan matricát, hogy „Vírusoknak és hackereknek belépni tilos”, de akkor hogyan is röhöghetnénk nagyokat, hogy még ez az óriáscég is annyira felületes (képzetlen?), hogy nem fordít figyelmet a hacker és a cracker – azaz a szakember és a kalóz – közötti különbségre…

---

betörő (cracker) – programok, rendszerek stb. feltörését végző, általában rosszindulatú vagy törvényekbe ütköző céllal tevékenykedő ember.
betyár (hacker) – általában magasan képzett, biztonsági megoldásokban jártas rendszergazda vagy programozó, aki a törvénysértés szándéka nélkül foglalkozik a különböző rendszerek sebezhetőségével, biztonsági réseivel, képességeivel.
cracker → betörő, kalóz.
előugró menü → helyi menü.
felbukkanó menü → helyi menü.
hacker – betyár, szaki.
helyi menü – a grafikus felületek egy eleméhez tartozó menü, amelyet általában az egér jobb gombjával az adott elemen kattintva jeleníthetünk meg.
kalóz lásd: betörő.
kombinált lista → lenyíló lista
kombipanel → lenyíló lista
legördülő menü – némely nyersfordításban megtalálható furaság. (Egy menü egyébként szokása szerint lenyílik.) Általában a gombmenü (egy gombhoz tartozó menü, amely a gombra kattintással jeleníthető meg) helyett hibásan használt kifejezés.
lenyíló lista – a grafikus felület eleme, s mely egy beviteli mezőből és egy ikonból (általában egy lefelé mutató nyíl) áll, az ikon segítségével a beviteli mezőhöz tartozó értéklista jeleníthető meg.
monolithic → tömbös szerkezetű
monolitikus → tömbös szerkezetű (a monolit szó szerint terméskövet jelent)
robosztus → általában hibásan használt szó. Sőt szinte csak hibásan használják, főleg az erős, hatékony, egységes, terhelhető, üzembiztos szavak valamelyike helyett.
robust → robosztus
tömbös szerkezetű (monolithic) – olyan program, amely nem modulokra bontva készült, hanem egy nagy egységként létezik. Egy bizonyos méret felett a kód nehezen kezelhetővé válik, főleg csoportos fejlesztések esetén előbb-utóbb eljutunk az „ősállatprogram” állapotához (nagy, okos, és senki sem érti, hogy működik).
szaki lásd: betyár.

Beköszöntő



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

Előző számunkban igen komoly arányban a gyakorló rendszergazdáknak kedveztünk, ezt a „kilengést” szeretnénk most egy csöppet jóvá tenni. Természetesen, ez nem azt jelenti, hogy a májusi számban nem talál érdekességet egy gyakorló szakember! A programozóknak kívánunk kedveskedni két fordító bemutatásával is (a GCJ a 39. oldalon, az Intel fordítója a 42. oldalon mutatkozik be), és természetesen a webes eszközöket bemutató Kovácsműhelyben is találhatunk két érdekes cikket. Sokan vannak, akik eltájékozódnak a GNU/Linux világában, de hálózati megoldásként még nem használták. Nekik szól Dobbantó rovatunkban három cikk is: ***Fülöp Balázs*** egy életképes levélkiszolgáló telepítését mutatja be (Postfix és Cyrus, 58. oldal), ***Garzó András*** a windowsos ügyfeleket kiszolgáló Samba rendszerbe enged betekintést (A Sambáról röptében, 62. oldal), ***Csontos Gyula*** pedig a hordozható gépek hálózati beállítását megkönnyítő módszert ismertet (Laptopnet, 64. oldal).

Hozzáfűzöm, mindig is fura dolog a kezdők számára, hogy egy feladatot olyan rengeteg különböző módon oldhatnak meg. Miért nincs egy egyszerű és hatékony megoldás egy ilyen általános feladatra, mint például a több hálózathoz is csatlakoztatható laptopok beállítása? Ha tovább gondolkodunk, rájövünk, hogy az, ami az egyik felhasználónak hatékony, a másiknak csak nyűg. A legjobb talán az volna, ha egy Vállalkozó Szellem felvállalná, hogy ír egy mindenki igényét kielégítő programot (úgy, hogy az minden Linux alatt működjön), majd karbantartja azt. Igen ám, de általában az a helyzet, hogy több V. Sz. is jelentkezik egyszerre, mindegyik elkezdi a saját kis világmegváltó programját, mindegyik programnak lesznek jó és rossz tulajdonságai, erősségei, hiányosságai. A világ pedig majd választ, hogy „melyik a használható”. Rengetegszer találkozunk olyan programokkal, melyek korábban különálló csomagok összevonásából jöttek létre, vagy a fejlesztésüket leállították, mert egy másik csomag tudásban felülmúlta, versengeni pedig értelmetlen volna. Ez tehát a nyílt fejlesztések menete. Igaz, hogy sokszor párhuzamosan fejlődnek azonos feladatú programok, de vagy van rájuk igény, vagy idővel átadják a helyüket ügyesebb testvéreiknek. És néha előfordul, hogy nagyon sokáig dolgoznak egymás mellett hasonló képességek kifejlesztésére a csoportok. Nem kell messze mennünk példáért, ott vannak a felhasználói felületek. Szomorúan csóváltam a fejem, amikor linuxos szakemberek szinte ölre mentek, például a KDE vagy a Gnome érdekeit védve. Még inkább elszomorított, mikor rendszeresen találkoztam olyan programokkal, amelyek csak KDE vagy csak Gnome alatt voltak képesek (hajlandóak?) futni. Ezt a témakört kívánja körbejárni ***Marco Fioretti*** a 20. oldalon kezdődő „A nagy egyesített munkafelület” című cikkében. De ezek a versengések eltörpülnek a fizetős termékek közötti kíméletlen verseny mellett, gondoljunk csak a komoly pénzekbe kerülő vállalatirányítási rendszerekre! Tudom, tudom, már többször szóltam ezzel kapcsolatban, így nem kell részleteznem rózsaszín jövőképemet. Szerencsére nem vagyok egyedül az elképzelésemmel, és – talán egy kicsit elhamarkodva, de – már közel érzem az időt, amikor egy linuxos szakember a „fájlkiszolgáló, FTP-kiszolgáló, adatbázis-kiszolgáló, webkiszolgáló telepítése” mellé ugyanilyen könnyedén odaírja: „átfogó vállalatirányítási rendszer telepítése”. Aki nem hiszi, olvassa el a 10. oldalon kezdődő cikket!

Jó Linuxozást kívánok!

Programvadászat

Havi korongunkon a legterjedelmesebb rész a SuSE Linux 8.2-es vadonatúj kiadásának egy CD-s telepíthető változata. Erről a változatról a 74. oldalon kezdődő SuSE Linux 8.2 Professional cikkünkben kaphatunk teljesebb képet; itt csak a saját élményeimet írom le.

SuSE 8.2

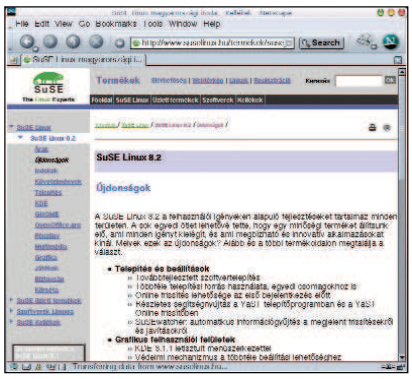
A telepítés a szokásos módon zajlik, azonban néhány érdekességre felhívnám az olvasók figyelmét.

Merevlemez-felosztás

Járjunk el körültekintően a merevlemez felosztásánál! Az előző SuSE megjelenésekor is írtam, hogy a rendszer önműködően a legnagyobb lemezrészemet akarta használni, ami nekem természetesen nem tetszett, ezért módosítanom kellett a telepítő elképzeléseit. Nagyon kellemes azonban, hogy a telepítő most már az NTFS lemezrészek „kurtítására” is képes, így nem okozhat gondot, ha esetleg Windows XP-t vagy 2000-et, esetleg NT 4-et futtatunk a gépünkön. Nem kell a teljes merevlemezt újra felosztanunk, így elkerülhetjük Windowsunk újratelepítését.

- *A grafikus felület beállítása*
  A próbához használt gépben egy kétfejes Matrox G400-as grafikus kártya volt. Ezt a telepítő rendesen felismerte, azonban mindkét fejet 800×600 VESA üzemmódra állította be, s a második fejet nem sikerült eltávolítanom a telepítés során, csak később, miután a rendszer teljes telepítését követően a gépet újraindítottam. A másik érdekesség a monitor beállítása volt: ugyan az adatbázisban rábukkantam a monitoromra (úgyhogy akár magától is megtalálhatta volna), de önműködően nem ismerte fel. A megoldást az jelentette, hogy hagytam a telepítőt, hadd tegye a dolgát; majd amikor mindennel elkészült és bejelentkezhettem az élő rendszerbe, gyorsan mindent a kedvem szerint állítottam át.
- *Programarzenál*
  A KDE 3-nak és az egyéb kiegészítőknek köszönhetően mindenre találhatunk grafikus alkalmazást; nekem mint megrögzött karakterfelület-bűvölőnek nagyon hiányzott a Midnight Commander. Sajnos ezt sem a teljes, sem pedig a korongunkon lévő változat nem telepíti fel alapértelmezésként. Természetesen később bármikor telepíthetjük.
- *A teljes változat telepítése*
  Mivel CD-ről telepítettem, mind az öt CD-t előkészítettem. Elindítottam a rendszert az első korongról, majd figyeltem, mi is történik.
  A telepítés rendben lezajlott, azonban számomra egy kissé érthetetlenül mind az öt CD-t bekérte tőlem. Nos kérem szépen, nem mondhatnám ezt a dolgot a csomagszervezés magasfokának, ugyanis a hármas-négyes-ötös korongról mindössze 10 MB – 400 KB – 18 MB adatot kellett feltenni. Ezeket a csomagokat szerintem nyugodtan a második, esetleg a harmadik CD-re tehették volna, ezzel is egy kicsit felhasználóbarátabbá téve ezt a különben igen jó rendszert. Ha valaki DVD-ről telepít, akkor ezt a macerát elkerülheti.
- *Összegzés*
  Nagyon könnyen telepíthető Linux-változatról van szó, azt a kis kellemetlenséget a Matroxommal leszámítva, a számítógépemnek minden alkatrészét felismerte. Nem kellett CompactFlash-olvasómmal sem bajlódnom, gond nélkül használatba vehettem.

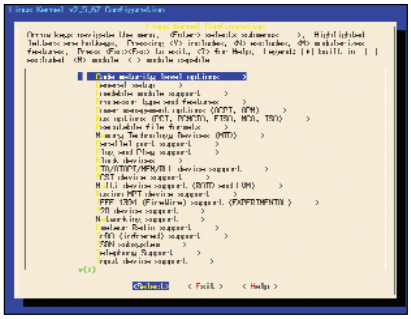
Aki a mellékelt korongról telepíti a rendszert, teljes értékű Linuxot kap; és amennyiben gyors internetkapcsolattal rendelkezik, akkor az Internetről az összes csomag elérhetővé válik a számára, így a teljes programkínálatból kedvére válogathat. Akik még nem kötelezték el magukat egy bizonyos Linux-változat mellett, azoknak mindenképpen ajánlom kipróbálásra; ki tudja, lehet, hogy ő lesz a nagy Ő.



Bővebb adatok:
➲ http://www.suselinux.hu/termekek/suse_linux/suse_82/

Magazin

A magazin cikkeihez kapcsolódó anyagaink a cikkekben feltüntetett helyen találhatók.



A talán legérdekesebb anyag *Marcel Gagné* cikkéhez kapcsolódó „asztalmegosztó” programok. A KDE-hez tartozó program a fentebb említett SuSE-kiadásban is megtalálható, így ha valaki kedvet érez hozzá, kipróbálhatja a telepítés után. Felkerült a legújabb fejlesztői rendszermag is, amivel bárki kedvére ismerkedhet. Szükség is van erre, mivel hamarosan (reményeim szerint) megjelenik a 2.6-os megbízható mag, aminek ez az elődje. Az első magyarországi PHP Konferencia március 29-én került megrendezésre. Az előadások anyagai PowerPoint és OpenOffice.org formátumban is felkerültek.

- ***Hojtsy Gábor*** – Bevezetés a PHP világába
- ***Bártházi András*** – Portálok kialakítása, felépítése, biztonsági követelmények
- ***Szabó Dénes*** – A Smarty sablonrendszer bemutatása
- ***Palócz István*** – PHP-oktatási tapasztalatok
- ***Papp Győző*** – „Nagyban utazunk”

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

Mindenre kiterjedő Oracle-támogatás

Az Oracle és a UnitedLinux bejelentése szerint az Oracle termékeket UnitedLinux alapokon futtató ügyfelek ezentúl közvetlenül az Oracle részéről kaphatnak támogatást az operációs rendszerrel összefüggő kérdéseikkel kapcsolatban. Természetesen a támogatás igénybe vételéhez mind az Oracle-nek, mind az operációs rendszer készítőjének – Connectiva, SCO, SuSE, TurboLinux – támogatási szerződést kell kötnie az ügyféllel, de a hirtelen felbukkanó hibák megoldása felgyorsulhat, ami adott esetben jelentősen hozzájárulhat a károk mérsékléséhez.

Webkamera nagyoknak

A JVC GY-DV300 Streamcorder nevű kamerájának érdekessége, hogy egyszerre képes az általa látottakat szalagra rögzíteni és – QuickTime 6 alól, MPEG-4 formátumban – Interneten sugározható médiafolyamként a számítógépnek átadni. Ugyan maga a kamera már korábban – február óta – kapható volt, a QuickTime-támogatás egy most megjelent hálózati csomaggal vált elérhetővé. A kiegészítő egység a kamera aljára csatlakozik, és egy hálózati vagy flashmemória-kártya fogadására képes. A kiegészítő egy webkiszolgálót is futtat, amelyen keresztül a felhasználó távolról is teljes mértékben átveheti a kamera kezelését.

➲ http://www.jvc.com/pro

Sorosít az Adaptec

Az Adaptec több Serial ATA vezérlőt is bemutatott, amelyek egy része alaplapra építhető megoldás, míg mások különálló bővítőkártya formájában érhetők el.
A SCSI-vezérlők piacán messze vezető cég elsősorban a költségérzékeny vevőket próbálja termékeivel megcélozni; soros ATA RAID-vezérlőivel viszonylag olcsó merevlemezekkel is nagyteljesítményű, az adatokat biztonságosan őrző tárrendszerek építhetők.
A most megjelent 1210SA vezérlő két merevlemezt tud kezelni, ezeket RAID 0 vagy RAID 1 módban tudja használni. A vezérlő ugyanazt a HostRAID megoldást alkalmazza, mint amit a cég Ultra320-as SCSI-vezérlői. A később megjelenő vezérlők RAID 1/0 és RAID 5 tömböket is kezelnek majd, illetve nemcsak kettő, hanem négy-, nyolc és tizenhat kapus eszközöket is találunk majd közöttük.

➲ http://www.adaptec.com

A legcikisebb helyzet az életemben…

A Business 2.0 magazin immár harmadik alkalommal állította össze a „101 leghülyébb pillanat az üzleti életben” című listáját. Bár a szerkesztők inkább csak az amerikai érdekeltségű vállalatok termékéből válogatnak, úgy tűnik, így is bőven van miből szemezgetniük. Néhány gyöngyszem:

- Egy fejlesztői konferencián ***Brian Valentine***, a Microsoft vezető alelnöke kijelenti, hogy az összes operációs rendszer ugyanolyan… Mi meg csak szívunk.
- ***Barclay Knapp***, az NTL telekommunikációs cég vezetője úgy véli, hogy a 23,4 milliárd szép nagy szám, a nagy számokat pedig mindenki szereti. Csak arról feledkezett el, hogy ez mínuszban értendő: cége közel ennyi adósságot halmozott fel, amikor végül csődbe jutott.
- Egy gondban lévő romániai autógyár dolgozói úgy gondolták, hogyha mindnyájan elmennek egy spermabankba, ahol adakoznak, majd az így szerzett keresetüket felajánlják a cégnek, tehetnek valamit a vállalat húszmilliós adósságának a ledolgozásáért. Valaki azt is kiszámolta, hogy fejenként 400 alkalommal járva a spermabankban, valóban eltűnhetne a kérdéses adósság.
- A Mattel a Harry Potter termékcsalád részeként egy vibráló seprűt is piacra dobott. Egy 32 éves anyuka visszajelzéséből hamarosan kiderült, ő maga is legalább annyira élvezte a seprűn való lovaglást, mint hétéves kisfia, csak az elemek tartóssága nem elégítette ki az igényeit. A cég azóta visszavonta a terméket.

A teljes lista a magazin honlapján olvasható.

➲ http://www.business2.com

Letölthető az OpenOffice 1.1b

Az OpenOffice 1.0 megjelenésével nem állhatott meg a világ, az 1.1-es változatban számos újdonság mutatkozik be. Csak néhány ezek közül: PDF, Macromedia Flash és egyéb új formátumok megnyitása és mentése; a thai, hindi, arab és héber nyelvek támogatása; dokumentum küldése elektronikus levélhez csatolva, PDF formátumban; mobileszközök és formátumaik támogatása; XML és XHTML mentése; fogyatékosokat segítő szolgáltatások; továbbfejlesztett súgó. A csomag a megszokott operációs rendszerekhez tölthető le.

➲ http://www.openoffice.org

Hálózati mindenesek MontaVista Linuxszal

Két újszerű termékben jut szerephez a MontaVista Linux. A mindössze két éve megalakult Zultys Technologies két hálózati – vagy ha úgy tetszik, összes – termékében beágyazott Linux fut. A többféle célra is használható készülékek nyílt szabványok alapján működnek, belsejükben IBM PowerPC 440GP processzor található. A ZIP 4×4 egy üzleti kategóriás, egyszerre négy hívást kezelni képes IP-telefon és négykapus, felügyelhető ethernet hálózati kapcsoló egyben, de használható számológépként, illetve beépített telefonkönyvvel is rendelkezik. Az MX1200 két egység magas készülék; video-, fax- és hanghívások, adatkapcsolatok kezelésére egyaránt alkalmas, egyszerre használható telefonos alközpontként, hálózati kapcsolóként, útválasztóként és internetes átjáróként – mindezeket a szolgáltatásokat egyetlen felületről lehet felügyelni. A ZIP 4×4 400 dollárba kerül, míg az MX1200 ára kiépítéstől, a támogatott felhasználók számától függően 20 és 212 ezer dollár között alakul.
(A képen a ZIP 4×4 IP-telefon és hálózati kapcsoló látható.)

➲ http://www.zultys.com

Erősít az SMC

Az SMC bemutatta EliteConnect High Power családjának első tagját, egy 802.11b szabvány szerinti, 2,4 GHz-en üzemelő, nagy energiájú, vezeték nélküli hálózati kártyát. A kártya érdekessége, hogy a legtöbb terméknél jóval nagyobb, 200 mW-os teljesítménnyel továbbítja a jeleket – amivel nemzetközi forgalmazásának reménye alighanem szertefoszlott, hiszen csak az Egyesült Államokban szabad 500 mW-ig emelni a 802.11-es eszközök teljesítményét, a nemzetközi teljesítményhatár 100 mW. A kártya – emelt teljesítményének köszönhetően – kül- és beltérben egyaránt jobb minőségű, illetve nagyobb, eszményi körülmények között akár 900 méteres távolságról is használható összeköttetést biztosít. A rádiós kapcsolat a meglévő 64/128 bites WEP-titkosítással védhető, de a második negyedévben – egy frissítés letöltése után – a Wi-Fi Protected Access (WPA) támogatása is elérhetővé válik. A kártya 140 dolláros áron vásárolható meg, és az SMC tervei szerint hamarosan további nagyteljesítményű hálózati elemek is társulnak mellé.

A Cobol élt, él…

A theKompany.com elkészült a Kobol 1.3.2-es változatával. Az ANSI Cobol teljes mértékben megfelel az előírásoknak, az ANSI-bizottság által csak nemrég jóváhagyott, objektumközpontú kiterjesztéseket is megvalósító Cobol-környezet az első ilyen képességekkel bíró termék. A beépített fejlesztői környezettel, CVS-támogatással és egyéb, a fejlesztői munkát megkönnyítő szolgáltatásokkal is rendelkező, Linux, Mac OS X és Windows alá egyaránt elérhető Kobol a maga hatvan dolláros árával alighanem a legolcsóbb hasonló eszköz.

➲ http://www.thekompany.com/products/kobol

Lendületben a SuSE

A Ximian Inc. és a SuSE Linux AG közös bejelentése szerint a SuSE Openexchange levél- és csoportmunka-kiszolgálójához hamarosan a Ximian IS alkalmazása társul majd ajánlott felhasználóoldali alkalmazásként. A felhasználók Evolution alól is ugyanazokat a szolgáltatásokat érhetik majd el, mint Microsoft Outlook alól. Mivel az Evolution felülete, megjelenése kísértetiesen emlékeztet az Outlookéra, a felhasználóknak az új programra történő átszoktatása nem jelenthet gondot. A közös Ximian Connector for Openexchange megoldás valamikor a harmadik negyedévben lesz elérhető. A két cég további együttműködést is tervez, amelynek keretében ez év második negyedévében a Ximian – Red Hat Linux alatt már elérhető – Red Carpet Exterprise felügyeleti eszköze elkészülne a SuSE-terjesztésekhez is, jelentősen megkönnyítve a SuSE-rendszergazdák számára a programok központi terjesztését, telepítését és működésük figyelését.

TeraGrid

A SuSE Linux Enterprise Server 8 Itanium processzorokra szabott változatára épül az amerikai kutatókat szolgáló

TeraGrid tudományos rendszer. A National Science Foundation vezérletével létrehozandó TeraGrid összekapcsolt fürtök hálózata lesz. Segítségével több ezer kutató férhet hozzá számítástechnikai erőforrásokhoz, és végezhet időjárásmodellezést, biológiai és egyéb jellegű számításokat. Az Itanium 2 alapú fürtök telepítését az IBM Global Services végzi, összesen négy helyszínen. A hálózat 600 TB adat tárolására lesz képes, a tárolórendszer vezérlését az IBM programjai végzik majd.

SuSE Linux Enterprise Server 8

A SuSE Linux Enterprise Server 8 Itanium processzorokhoz készült változata március vége óta bárki által megvásárolható, a több CD-lemeznyi program és leírás mellé 12 hónapnyi karbantartás-szolgáltatás is társul.

FireWire-ethernet házasság

A 1394 Trade Association és az IEEE újonnan létrehozott közös csoportjának feladata olyan módszerek kidolgozása, amik IEEE 1394b összeköttetések létrehozását teszik lehetővé akár száz méter hosszúságú Cat5-ös vezetékeken, a meglévő Gigabit ethernet kapcsolatok felett. A csoport munkája eredményeképpen a FireWire-szabvány következő frissítésében már az újfajta kiterjesztés is szerepelhet. A fő feladat olyan végpontok és köztes kapcsolók, elosztók létrehozása, amelyek képesek választani a 10/100/1000 Mb/s sebességű ethernet, illetve a 400/800 Mb/s FireWire kapcsolatok között. Ilyen eszközökkel ethernet – FireWire kapcsolatokat is létre lehet majd hozni, így eltérő eszközök is csatlakozhatnak majd azonos alhálózatra. A felsőbb szintű protokollok számára mindez természetesen teljesen átlátszó módon fog történni.

Minősítési programot indít a Red Hat

A Red Hat új programot indított a független alkalmazásfejlesztők számára. A különféle fejlesztők részéről – állítják a Red Hatnél – egyre nagyobb igény mutatkozik arra, hogy programjaikat Red Hat Enterprise Linux alatt próbálhassák ki, és a megfelelő működésről minősítést szerezhessenek. A felmérések szerint a fejlesztők 59 százaléka tervezi, hogy jövőre valamilyen terméket fejleszt Linux alá. A Red Hat Ready program révén a Red Hat ügyfelei bizonyosságot szerezhetnek arról, hogy a minősítéssel ellátott alkalmazások gondos tesztelésen mentek keresztül, és fejlesztőik betartották a Red Hat által számukra előírt irányelveket. Az elvárásokat teljesítő alkalmazások felkerülnek a Red Hat hivatalos alkalmazáslistájára is, illetve tulajdonosaik termékük hirdetésekor jogosulttá válnak a Red Hat-logó használatára.

➲ http://www.redhat.com/partners

Serial ATA lassan nekünk is

A Seagate az első olyan merevlemezgyártó, amelynek az egyik Serial ATA merevlemeze – egy 120 GB-os, 7200-as fordulattal üzemelő folyadékcsapágyas egység – kiskereskedelemben is megvásárolható (hivatalosan a Best Buy oldalain), de a magyarországi kereskedések is fogadják már a rá vonatkozó előrendeléseket. A soros ATA merevlemezekről eddig inkább csak hallhattunk, Japánban a szerencsések vásárolhattak is ilyet, ám a magyar üzletek árlistáin nem akartak feltűnni az új meghajtók. A Seagate lemezének bruttó ára jelenleg ötvenezer forint körül jár, ami kedvezőnek semmiképpen nem mondható; ám mivel az

újdonságok ára mindig magas, néhány hét múlva valószínűleg jóval olcsóbban szerezhetjük meg. A soros kábellel csatlakoztatott meghajtók 150 MB/s átviteli sebességre képesek, az adatok épségét 32-bites CRC-vel védik, és a kábelezésük is messze kényelmesebb a jelenleg használatos szalagkábeleknél.

➲ http://www.seagate.com

Új Mozilla-útiterv jelent meg

A távoli múltba vész 1998 vége, amikor a mozilla.org keretein belül megkezdődött a Geckónak keresztelt böngészőmotor fejlesztése. A munka végül is haladt, csak alig győztük kivárni, míg a programozók abba az állapotba juttatták, hogy végre odafűzhették az 1.0 jelölést a Mozilla név mögé. Azóta rengeteg fejlesztés történt, és hamarosan életre kel az 1.4-es ág, ami az 1.0 leváltására hivatott, és viszonylag hosszabb ideig megbízható ágként szolgál majd. Az 1.5-ös és 1.6-os változatokban várhatóan merészebb lépésekre is elszánják magukat a fejlesztők. Tennivaló akad bőven, a kódra ráfér némi csinosítgatás, az XML-támogatással is haladni kell – ennek ellenére a cél nem a munka gyorsasága, hanem a minőség fokozása. Forradalmi újdonságokat tehát ne várjunk – az 1.4-es változat május, míg az 1.5-ös augusztus közepén készül majd el.

➲ http://www.mozilla.org

Törjünk Xboxot!

Aki megunta az RC5 kulcsok keresését, valamivel hasznosabb, ámbár kevésbé tiszta kezdeményezéshez is csatlakozhat: az Operation Project X célja a Microsoft által az Xbox adathordozók és programok aláírására használt 2048 bites RSA kódolás megtörése.
A szívet melengetően reménytelen elosztott tervezethez Linux, Windows, FreeBSD, NetBSD és MacOS alá, sőt Xbox-gépekhez egyaránt találunk ügyfélprogramot.
A Microsoft kódjának megtörésére azért van szükség, mert az Xbox csak az aláírt programok futtatását engedélyezi.
A védelmet modlapka beszerelésével ki lehet iktatni, ám ebben az esetben a cél az, hogy érintetlen gépen is lehessen például Linuxot futtatni. A Project X valójában az Xbox Linux tervezethez kötődik, amelynek keretein belül egy százezer dolláros díjazású „pályázatot" írtak ki olyan jogtiszta, mások számára ésszerű költséggel megvalósítható, könnyen megérthető és szabadon hozzáférhető megoldások készítésére, amelyek modlapka nélkül is lehetővé teszik más operációs rendszer telepítését Xboxra. Mivel első nekifutásra senki nem tudta elvinni a pénzt, a határidőt ez év végéig tolták ki.
➲ http://www.operationprojectx.com

Hangminősítési program számítógépekhez

Az nVidia és a Dolby Laboratories SoundStorm néven közös hangminősítési programot hozott létre alaplapok és személyi számítógépek számára.
A pályázó termékeknek gazdag szolgáltatáskészletet, élethű hangzást és magas teljesítményt kell nyújtaniuk – minő meglepetés, hogy elsőként nVidia nForce2 lapkakészletére épülő alaplapok teljesítik a követelményeket; bár a minősítésre méltó termékeknek természetesen nem feltétlenül nVidia lapkakészletre kell épülniük. A minősítendő eszközöknek különálló elülső jobb/bal, hátulsó jobb/bal és közép/mély analóg kimenettel, különálló S/PDIF kimenettel, különálló mikrofon- és vonali bemenettel kell rendelkezniük, analóg kimeneteiken 85, bemeneteiken pedig 80 dB jel/zaj értéket kell felmutatniuk. Ha mindezeket az elvárásokat teljesítik, akkor gyártójuk elhelyezheti rajtuk az nVidia- és Dolby-Digital logókat, mi pedig elfelejthetjük azt a régi nézetet, miszerint számítógépből kizárólag zajokkal terhelt, a vájt fülűek számára élvezhetetlen hang jöhet ki.

Tenyérfelismerés a Fujitsutól

A Fujitsu Laboratories egy érintést nem igénylő biometriai azonosítóeszközt dolgozott ki. A készülék a fölé helyezett tenyérben az erek futását követi és ismeri fel, és természetesen tartozik hozzá egy program is, ami a beolvasott adatokat dolgozza fel. A leolvasott képben az erek fekete vonalakként jelennek meg, a program ezt hasonlítja össze a tárolt mintákkal. A készülék viszonylag érzéketlen arra, ha alkalmanként változik a leolvasó és a tenyér közötti távolság, illetve ha az azonosított személy más-más szögben tartja a kezét a leolvasó fölé. A fejlesztők bevallása szerint megoldásuk mindössze az esetek egy százalékában utasít el valakit tévesen, és kétszáz esetből csak egyszer fordul elő olyan, hogy jogosulatlan személyt is felismerni vél.
A tenyérben futó erek – hasonlóan az ujjlenyomathoz – egy életen át megőrzik rajzolatukat, legfeljebb a minta mérete változhat meg. Az ujjlenyomat-leolvasók általában érintéssel működnek, ami viszont nagyobb esélyt ad a fertőzések terjedésére. A Fujitsu szakemberei különféle területeken ki szeretnék próbálni a készüléküket, illetve számos eszközbe tervezik a beépítését.

Xerox asszisztens gyengén látóknak

A Xerox új alkalmazásának segítségével vak és gyengén látó felhasználók is elboldogulhatnak egy digitális másológéppel. A Xerox Copier Assistant program személyi számítógépen fut, mellé már csak egy Document Centre 500-as többfeladatos gépre van szükség – mások segítségére nem.
A korszerű irodai gépeken általában kisméretű érintőképernyő szolgál a vezérlésre, csakhogy ezt nem mindenki tudja használni. A 495 dolláros Xerox Copier Assistant a számítógép képernyőjén, kinagyítva jeleníti meg a készülék felhasználói felületét, képes a menüpontok felolvasására, a műveletek elvégzését pedig a számítógép billentyűzetén keresztül teszi lehetővé. A Xerox a jövőben – megjelenésükkel összhangban – újabb termékeinek támogatását is beépíti a programba.

Medgyesi Zoltán (mz@rettesoft.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 hírszerkesztője.

Megoldás a középvállalatok legnagyobb gondjára

Már hosszabb ideje foglalkoztat a kérdés, hogy vajon létre tudunk-e hozni olyan nyílt vállalatirányítási rendszert, ami egy csapásra megoldaná a Linuxot használó középvállalatok tengernyi nehézségét. Tudom, minden vállalatnak egyéni igényei vannak, sajátos környezet és így tovább, tehát nehéz egy általános rendszert létrehozni. De találkoztam néhány érdekes kezdeményezéssel, ezeket szeretném most röviden bemutatni.
Először is gondoljuk végig, hogy mi lenne a legjobb megoldás! Szét kell választani a kis-, a közép- és a nagyvállalatok igényeit. Úgy érzem, hogy az igazán nagy hiány a piacon a középvállalatok területén mutatkozik, hiszen a nagyvállalatok könnyedén kifizetnek tíz- vagy százmilliókat a nagyrendszerekért, a kisvállalkozások teljes ügyintézését pedig gyakran egyetlen munkatárs el tudja látni. A kettő közötti méretnél már szükség van a „nagyok" tudására és pontosságára, azonban gyakran nincs ráfordítható néhány tízmillió forint. Keresünk tehát egy olyan rendszert, ami megbízható, elérhető árú (vagy lehetőleg ingyenes), ráadásul a fenntartási és tulajdonlási költségei alacsonyak.

Felhívás

Több éve üzemelő vállalatirányítási rendszer keres közepes méretű vállalatokat, ahol a tudását kamatoztathatná. A felmerülő költségek fedezéséhez minisztériumi forrásokból támogatás pályázható. Kizárólag a nyílt forrású megoldásokat kedvelő cégek jelentkezését várjuk!
Ugyanitt keresünk lelkes programozókat, akik szívesen áldoznak idejükből egy országszerte használható nyílt rendszer moduljainak átnézésére, továbbfejlesztésére, dokumentálására. Előny, ha a jelentkező a későbbiekben örömmel vállal megbízást a rendszer bevezetésére, betanítására.

Egy-egy ilyen közepes vállalatnál találhatunk egy helyi hálózatot a legkülönbözőbb gépekkel, egy számlázóprogramot (például saját fejlesztést, esetleg valamilyen külső cégtől vásárolt, de gyakran az igényeknek nem tökéletesen megfelelő rendszert), egy könyvelőprogramot, ami nem képes együttműködni a számlázóprogrammal, és egy rendszergazdát, aki rugdossa a programokat, ha nem akarnak működni (emellett fejből tudja a könyvelőprogram fejlesztőjének telefonszámát).
Most induljunk ki abból, hogy a cég végre egy komoly vállalatirányítási rendszert szeretne, túl akar lépni az idegtépő, gyakran heteket igénybe vevő informatikai küszködésen. Mit tehet a cég, ha rájön, hogy a három út közül (saját fejlesztés, kész rendszer vásárlása, nyílt forrás használata) hosszú távon a harmadik a gazdaságos? (Aki nem erre a végeredményre jutott, szívesen meghallgatom a véleményét, számomra ugyanis hosszú távon egyértelmű a nyílt rendszerek fölénye.)

Nemzetközi programok

Két komolyabb ERP/CRM megoldással találkoztam, amelyek ugyanezt a célt tűzték ki; az egyik a GNU Enterprise, mely a GNU égisze alatt fejlődik, a másik pedig a régebben független Compiere, ami ma már a Sourceforge.net egyik legaktívabb csoportja. Mindkettő igen ígéretes, mindkettőben rendkívül sok jó megoldás van, azonban egyikhez sem létezik magyar felület (tudtommal). Talán éppen ez a két rendszer leggyengébb pontja. Egy középvállalat, ahol minden forint számít, nem egykönnyen helyezi a teljes vállalatirányítását olyan termékre, amihez legfeljebb külföldről kaphat segítséget. Vállalatvezetőként csupán két jó megoldást látok, ami hosszú távon biztosítja számomra a rendszer használhatóságát. Fontos ugyanis, hogy a cég huzamosabb ideig ne függjön egy külső (főleg egy nehezen elérhető) társaságtól. Az első megoldás, ha van egy munkatárs a cégnél, aki házon belül meg tudja oldani a kérdéseket, a második pedig, ha egy külső cég elérhető áron bármikor segítséget tud nyújtani a rendszerrel kapcsolatban. Sőt jó esetben nem egy, hanem sok ilyen cég közül is választhat a vállalat.
Az említett két rendszer tehát csak akkor jönne szóba, ha létezne olyan tanácsadó és informatikai szolgáltató cég Magyarországon, ami ezeket a rendszereket a magyar jogszabályokhoz illeszti, beüzemeli és be is tanítja, illetve a karbantartásról sem feledkezik meg. Ebben az esetben csupán a cég szolgáltatásáért fizetnénk, ami már korántsem tűnik olyan vérlázítónak, mint amit egy-egy fejlesztőcég követési díj vagy éppen használati engedély címén elkér. Sajnos ilyen, az előbbi fajtából való cégről nem tudok, tehát marad az a megoldás, hogy a vállalat fenntart egy embert a rendszer megismerésére, beüzemelésére és karbantartására.

Egy működő rendszer

Korábbi cikkeim kapcsán *Gibizer Tibor* révén kerültem kapcsolatba *Kozák Kálmán*-nal, aki a TRON rendszer útját egyengeti. Ez a vállalatirányítási rendszer már hosszú évek óta sok cégnél folyamatosan bizonyít – kereskedelmi és szolgáltató cégeknél egyaránt. A rendszer fejlesztői pont azt célozták meg, amit az előző cikkemben leírtam: egy teljes mértékben Linux alatt működő, nyílt eszközöket és rendszereket használó hálózati, közepes méretű vállalatok szinte minden igényét kielégítő rendszerről van szó, ami PostgreSQL-t használ adatbázismotornak.
Nagyot dobbant a szívem, amikor megtudtam, hogy a TRON rendszert csupán egyszer kell megvásárolni, nincs követési díj, nincs éves kötelező karbantartás stb.
Beszélgetésünk során Kálmán elmondta, hogy a hosszú évek óta üzemelő moduláris rendszert egy lelkes csapat hozta létre, és nagyon sokat köszönhetnek a nyílt forrású rendszereknek. Ezért elhatározták, hogy valahogy „visszafizetik" a nyílt világnak ezt a rengeteg támogatást. Hogyan is lehetne visszafizetni? Természetesen a kód megnyitásával, közzétételével. Tehát itt van egy rendszer, ami tudja a szükséges szolgáltatásokat, és még a tulajdonosnak is az a célja, hogy megnyissa a kódot! Remek! De miért nem történt meg eddig a megnyitás? Sajnos a kis csapat számára erőn felüli feladatot jelent a kód olyan szintű rendbe tevése, hogy azt nyugodt szívvel

a világ elé tárják. A „gatyába rázás” a jelenlegi felállás szerint akár több évig is eltarthat, ezt pedig a cég nem engedheti meg magának. Kérdeztem, mit szólna hozzá a csapat, ha többen beszállnánk a munkába, mind cégek, mind fejlesztők, akik szívesen áldoznak idejükből egy ilyen rendszerre. Morfondírozásunk eredményeként egy idilli folytatás alakult ki bennem: összejön négy-öt cég, akik ezt a rendszert kívánják használni, összedobják a bevezetésre amúgy is szánt összeg egy részét (mondhatnók, egy másik rendszer megvásárlására szükséges pénz töredékét), adják hozzá a saját cégükben amúgy is meglévő plusz egy főt, jelentkezik még néhány lelkes programozó, és pikk-pakk, szépen, modulonként átfésüljük a rendszert, egyúttal be is vezetve az adott modulokat a résztvevő cégeknél.
Természetesen komoly kihívást jelent egy teljes vállalatirányítási rendszer átfésülése, nem kisebbet a rendszer éles bevezetése. Ezzel együtt bizakodó vagyok, ugyanis egy jelentkező céget már most tudok, és egy maroknyi ember már jelezte is, hogy szívesen részt venne a munkában. Ha mindehhez hozzávesszük, hogy a Gazdasági és Közlekedési Minisztérium külön pályázati felhívást tett közzé „a kis- és középvállalkozások részére a korszerű vállalatirányítási rendszerek támogatására”, ahol akár négymillió forint támogatást is lehet nyerni, akkor még szebb a kép. Minisztériumi támogatással akár további költségek nélkül állhatnak át vállalatok egy korszerű és nyílt megoldásra. Ki tudja, az év végére talán magazinunk lemezmellékletén szerepelhet a teljes TRON. Úgy legyen!

Szy György (Szy.Gyorgy@linuxvilag.hu)
A Linuxvilág főszerkesztője,
a Kiskapu Kiadó vezetője.
Mindenki levelét örömmel várja.

KAPCSOLÓDÓ CÍMEK

GKM: Pályázati felhívás a kis- és középvállalkozások részére a korszerű vállalatirányítási rendszerek támogatására ➲ http://www.gkm.hu/site/fomenu/palyazatok/2003/uj/szvp/SZVP-2003-2.htm
TRON Vállalatirányítási Rendszer ➲ http://www.tron.hu/
Compiere ➲ http://www.compiere.org
GNU Enterprise ➲ http://www.gnue.org

Linux-index

1. Ekkora összeget ajánlott fel (és fizetett ki 2003 januárjában) *Michael Robertson*, a Lindows alapítója arra a célra, hogy a Linuxot Microsoft Xboxra ültessék át 2002 végéig: 100 000 dollár
2. Ekkora összeget ajánlott fel Michael Robertson arra a célra, hogy az átültetést eszközmódosítás nélkül hajtsák végre 2003 végéig: 100 000 dollár
3. Várhatóan ekkorra válik a Linux a kiszolgálók vezető operációs rendszerévé: 2009
4. Az a pozíció, amelyet a Linux várhatóan hamarosan megszerez az asztali gépek operációs rendszerei között: 2.
5. Ennyi rúpia az ára a Hewlett-Packard 1,5 GHz-es, AMD Athlon processzoros, Presario típusú személyi számítógépének Red Hat Linuxszal: 30 990 (645 dollár a 2003. január 3-i adatok szerint)
6. Ennyi rúpia az ára a Hewlett-Packard 1,6 GHz-es, Intel processzoros, Presario típusú személyi számítógépének Windows XP-vel: 40 000 (833 dollár a 2003. január 3-i adatok szerint)
7. További árkedvezmény rúpiában a linuxos Presarióra az „összeszerelőktől”: 2000
8. Ennyi millió találatot adott a Google a Linux szóra 2002. november 29-én: 41
9. Ennyi millió találatot adott a Google a Linux szóra 2003. január 2-án: 59
10. Naponta ennyivel nőtt a Linux-találatok száma a fenti időszakban: 529 412
11. Ennyiedik helyen áll a Linux a tíz vezető technológiára végzett Google-keresés eredménye alapján: 4.
12. Ennyiedik helyen áll a Microsoft a tíz vezető technológiára végzett Google-keresés eredménye alapján: 9.
13. Ennyi alkalmazást mutatott be a Sun Microsystems a Comdex Fall 2002 kiállításon: 2
14. Ennyi Linux-alkalmazást mutatott be a Sun Microsystems a 2002-es Comdex Fall kiállításon: 2
15. Várhatóan a kiszolgálók ekkora hányadán fut majd Linux 2006–2007-re: 45

Források

1–2.: Xbox Linux Project (➲ http://xbox-linux.sourceforge.net)
3.: a Butler Group által készített jelentés a kiszolgálókon futó operációs rendszerekről (➲ http://www.butlergroup.com)
4.: IDC (ZDNet-en keresztül)
5–7.: Financial Express
8–10.: Google
11–12.: Search Engine Watch
13–14.: *Doc Searls* jelentése a Comdex kiállításról
15.: Meta Group, megjelent a BusinessWeek magazinban

Linux Journal 2003. április, 108. szám

a világ elé tárják. A „gatyába rázás” a jelenlegi felállás szerint akár több évig is eltarthat, ezt pedig a cég nem engedheti meg magának. Kérdeztem, mit szólna hozzá a csapat, ha többen beszállnánk a munkába, mind cégek, mind fejlesztők, akik szívesen áldoznak idejükből egy ilyen rendszerre. Morfondírozásunk eredményeként egy idilli folytatás alakult ki bennem: összejön négy-öt cég, akik ezt a rendszert kívánják használni, összedobják a bevezetésre amúgy is szánt összeg egy részét (mondhatnók, egy másik rendszer megvásárlására szükséges pénz töredékét), adják hozzá a saját cégükben amúgy is meglévő plusz egy főt, jelentkezik még néhány lelkes programozó, és pikk-pakk, szépen, modulonként átfésüljük a rendszert, egyúttal be is vezetve az adott modulokat a résztvevő cégeknél.

Természetesen komoly kihívást jelent egy teljes vállalatirányítási rendszer átfésülése, nem kisebbet a rendszer éles bevezetése. Ezzel együtt bizakodó vagyok, ugyanis egy jelentkező céget már most tudok, és egy maroknyi ember már jelezte is, hogy szívesen részt venne a munkában. Ha mindehhez hozzávesszük, hogy a Gazdasági és Közlekedési Minisztérium külön pályázati felhívást tett közzé „a kis- és középvállalkozások részére a korszerű vállalatirányítási rendszerek támogatására”, ahol akár négymillió forint támogatást is lehet nyerni, akkor még szebb a kép. Minisztériumi támogatással akár további költségek nélkül állhatnak át vállalatok egy korszerű és nyílt megoldásra. Ki tudja, az év végére talán magazinunk lemezmellékletén szerepelhet a teljes TRON. Úgy legyen!

Szy György (Szy.Gyorgy@linuxvilag.hu)
A Linuxvilág főszerkesztője,
a Kiskapu Kiadó vezetője.
Mindenki levelét örömmel várja.

KAPCSOLÓDÓ CÍMEK

GKM: Pályázati felhívás a kis- és középvállalkozások részére a korszerű vállalatirányítási rendszerek támogatására ➲ http://www.gkm.hu/site/fomenu/palyazatok/2003/uj/szvp/SZVP-2003-2.htm
TRON Vállalatirányítási Rendszer ➲ http://www.tron.hu/
Compiere ➲ http://www.compiere.org
GNU Enterprise ➲ http://www.gnue.org

Linux-index

1. Ekkora összeget ajánlott fel (és fizetett ki 2003 januárjában) *Michael Robertson*, a Lindows alapítója arra a célra, hogy a Linuxot Microsoft Xboxra ültessék át 2002 végéig: 100 000 dollár
2. Ekkora összeget ajánlott fel Michael Robertson arra a célra, hogy az átültetést eszközmódosítás nélkül hajtsák végre 2003 végéig: 100 000 dollár
3. Várhatóan ekkorra válik a Linux a kiszolgálók vezető operációs rendszerévé: 2009
4. Az a pozíció, amelyet a Linux várhatóan hamarosan megszerez az asztali gépek operációs rendszerei között: 2.
5. Ennyi rúpia az ára a Hewlett-Packard 1,5 GHz-es, AMD Athlon processzoros, Presario típusú személyi számítógépének Red Hat Linuxszal: 30 990 (645 dollár a 2003. január 3-i adatok szerint)
6. Ennyi rúpia az ára a Hewlett-Packard 1,6 GHz-es, Intel processzoros, Presario típusú személyi számítógépének Windows XP-vel: 40 000 (833 dollár a 2003. január 3-i adatok szerint)
7. További árkedvezmény rúpiában a linuxos Presarióra az „összeszerelőktől”: 2000
8. Ennyi millió találatot adott a Google a Linux szóra 2002. november 29-én: 41
9. Ennyi millió találatot adott a Google a Linux szóra 2003. január 2-án: 59
10. Naponta ennyivel nőtt a Linux-találatok száma a fenti időszakban: 529 412
11. Ennyiedik helyen áll a Linux a tíz vezető technológiára végzett Google-keresés eredménye alapján: 4.
12. Ennyiedik helyen áll a Microsoft a tíz vezető technológiára végzett Google-keresés eredménye alapján: 9.
13. Ennyi alkalmazást mutatott be a Sun Microsystems a Comdex Fall 2002 kiállításon: 2
14. Ennyi Linux-alkalmazást mutatott be a Sun Microsystems a 2002-es Comdex Fall kiállításon: 2
15. Várhatóan a kiszolgálók ekkora hányadán fut majd Linux 2006–2007-re: 45

Források

1–2.: Xbox Linux Project (➲ http://xbox-linux.sourceforge.net)
3.: a Butler Group által készített jelentés a kiszolgálókon futó operációs rendszerekről (➲ http://www.butlergroup.com)
4.: IDC (ZDNet-en keresztül)
5–7.: Financial Express
8–10.: Google
11–12.: Search Engine Watch
13–14.: *Doc Searls* jelentése a Comdex kiállításról
15.: Meta Group, megjelent a BusinessWeek magazinban

Linux Journal 2003. április, 108. szám

Az első OpenOffice.org-konferencia

Az első OpenOffice.org-tanácskozást Hamburgban tartották, pont a hannoveri CeBIT utánra időzítve.

Kis hazánk képviseletében (az LME pénzügyi segítségével megtámogatva, amit ezúton is köszönünk) négyfős csapat kelt útra: *Varga S. Csaba (Guska)*, *Somogyi Péter (Jerry)*, *Noll János (JohnZero)*, valamint jómagam. A helyszínválasztás számunkra annyiban volt szerencsétlen, hogy 14 órás autóutat kellett legyűrnünk. (Odafelé, visszafelé viszont 16 órásra sikeredett.)
De sebaj, az OOo-t lánykorában – amikor még zárt program volt – Hamburgban kezdte fejleszteni a német StarDivision cég, és a legtöbb (ma már Sun Microsystems-színekben futó) fejlesztő ma is arrafelé található, úgyhogy talán megbocsáthatjuk nekik a dolgot.

A konferencia helyszínéül a hamburgi egyetem bölcsészkari épületének (Philoturm) egyes termeit választották.
Az alaphangot *Miguel de Icaza* adta meg, aki nyitóbeszédében az OOo világszerte kivívott fontos szerepéről szólt. A nyílt adatformátumok fontossága után a könnyű honosíthatóság témaköre került terítékre. (Miguel egy thai csapatot említett, akik már két hét alatt megcsinálták, bizony. Mi pedig úriemberként nem kiabáltunk közbe, hogy eldicsekedjünk a magunk három napjával.)
Az általános helyzetkép után Miguel rögtönözött kívánságlistába kezdett: a diái között össze-vissza ugrálva, a neten böngészve pár érdekes látványbeli szolgáltatást ajánlott (➲ http://www.smartmoney.com/marketmap/ és ➲ http://www.kartoo.org/), amit jó lenne valamikor viszontlátni az OOo-ban. Jó Gnome-osként a sok nyelven való programozhatóság és a makrózhatóság sem maradt le a listájáról.
Egy régi StarDivision-os motoros, *Jürgen Schmidt* az OOo fejlesztésének alapját képező UNO (Universal Network Objects) összetevő (component) keretrendszert ismertette. Pár egyszerű forráskódos, lefordítós, futtatós példával ott helyben szemléltette is, hogyan lehet az UNO javás illesztőfelülete révén a Javában szolgáltatásokat adni a nagy programokhoz – a Calc részbe pillanatok alatt megvalósított egy számlálóbabot (bean), meg egy cellaértékszámítási kifejezésekben felhasználható függvényt.
Erwin Tenhumberg (Sun) az OOo átalakuló fejlesztési házirendjéről beszélt: ezután nem az történik, hogy az összes kódoló egyből a CVS HEAD ágába tesz közzé (commiter) mindent, hanem az egyes részterületek – szolgáltatások, hibajavítások stb. – egy-egy

kérészéletű alágat (egy Child WorkSpace-t – CWS) nyitnak, és abban a maguk számára megnyugtató módon összehozzák a kérdéses kódot, majd utána folyatják vissza az egészet a főágba, a Master WorkSpace-be (MWS).
Az előadás vége felé annyi közönségkérdés érkezett, hogy egy másik sunos fejlesztő, *Martin Hollmichel* vette át a válaszadó stafétáját. Úgy tűnik, ezzel az átszervezéssel a Sun is ésszerűsíteni akar: megszüntetik a saját belső StarOffice-főágukat, a nyilvános OOo-repository lesz az ő alapjuk is, és a saját fejlesztéseiket csak ehhez képest rakják rá. A két fejlesztés közt így akár közös binárisok is elképzelhetők.

A CollabNet-es *Louis Suarez-Potts* az OOo-fejlesztők közösségének (tervezett) átalakulásáról beszélt: olyan szerkezetet szeretnének, amivel az egyáltalán nem hozzáértő, de segíteni akaró kívülállók is akár öt perc leforgása alatt a saját képességeiknek megfelelően tudnak segíteni. Ez nem egyszerű, ugyanis hogyan

lehet annyira átláthatóvá tenni a fejlesztők illetékességi köreit, hogy a kívülálló is eligazodjon benne, és a túl nagy erőfeszítés láttán ne vesszen el a jó szándék? És ne is lépjenek egymás lábára! A megoldás fontos lenne, mert pár százezer felhasználó kisméretű, párhuzamos hozzájárulásából sok minden megszülethetne. Lassú víz partot mos. Az összehangolás technikai hátteréről annyi hangzott el, hogy minden részprojekt fejlesztéstámogató programjának kapcsolatot kell tudnia tartani a CollabNet-es SourceCasttal, s ezáltal nagyobb konzisztencia érhető el például a kifelé látható webfelületen.

John M. Rice nagyon elegáns, mintaszerű előadást tartott az OOo parancsnyelvkezelő keretrendszeréről: egy közös alapba további nyelvi csatolókat lehet beilleszteni. Egy nyelvi csatoló egy új parancsnyelv számára elérhetővé teszi az OOo saját és a többi parancsnyelvből származó függvényeit és objektumait. A dokumentumokba menthető parancsfájlok tehát mindet egyformán, a saját parancsnyelvük írásmódjával használhatják. Maga a keretrendszer még nincs a letölthető OOo-csomagba bevonva, de az 1.1-ben már ígérik, addig is pár JAR-fájl letöltésével elérhető. A kiadásig néhány szálat még el kell varrni: nekünk főként a grafikus felület kérdése tűnt fel. A bemutatott és lefuttatott bemutatókban (demo) ugyanis a kód a javás Swing-felülettel jelent meg, ami igen elütött az OOo saját kinézetétől. A dokumentumok adatait kezelő részek attól még nagyon szépen futottak.

Chris Halls, a Debian-terjesztés egyik OOo-csomagkarbantartója (de ott volt *René Engelhart* is) röviden bevezette a Debian alapelveit, nagy hangsúlyt fektetve a szabad programokra. Bemutatta a terjesztés szerkezetét (main, contrib, non-free), majd elhelyezte benne az OOo-t (a contrib-ban van, mert a fordításához még szükséges a Sun-féle JDK). Szólt a csomagolás folyamatáról (foltozás, fordítás, telepítőkészlet, az OOo szokásos telepítése során lejátszódó `./setup` hívás közül az elsőt elkészítik, az eredményét csomagolják), a felhasználói visszajelzésekről (a Debian az i386-os kiépítésen kívül még S390-en is karbantartja az OOo-t). Végül sajnos egy szomorú hírrel szolgált: pár hónapja olyan hírek érkeztek, miszerint az OOo-fejlesztők megcélozzák a felületek (platforms) saját csomagkezelőinek használatát (az OOo saját telepítője helyett) – erről azóta sem hallott túl sokat, tehát nem érdemes rá lélegzet-visszafojtva várnunk.

Michael Meeks, Miguel mellett a konferencia másik jelentős ximianos fejlesztője – és az, akivel az első nap együtt ebédeltünk – a Gnome-mal történő egyesítésről tartotta nagy közönségfigyelemmel kísért előadását. Az eredmények valóban szemet gyönyörködtetőek: az egységesebb ikonoktól kezdve az egyszerűsített menüpontokig, a simított betűtípusoktól a felturbózott CUPS-támogatásig (nyomtatókezelés). Viszont hiába támogatja a Sun a Gnome-ot, az OOo GTK-ra áttérése még csak tervbe sincs véve. Különféle cégpolitikai okokból pedig még nem is adták ki a fejlesztéseiket, egyelőre a Ximian Desktop 2 (XD2) nevű termékükre tartalékolják, a nagyközönség előtti megjelenés időpontja nem ismert. Még egy érdekesség; egy kérdésre válaszolva Michael a Ximian Bonobo nevű komponensrendszerét éppen hogy csak nem illette az „elavult" szóval (pedig a nyelve hegyén volt), nagyon dicsérte viszont az UNO-t, úgyhogy valószínűleg érdekes fejlemények várhatók.

Ami minket érint: Noll János előadásában bemutatta a magyar OOo-honosító környezetét, és a bennlévő közönségtől meglepően sok kérdést kapott. Mi több, *Nils Fuhrmann* révén a program fel is került egy sunos gépre, majd azon melegében – szó szerint a konferencia alatt – megindult a török honosítás, és a tanácskozás végére (azt hiszem) be is fejezték! Nilsszel beszélgetve bukkant fel a hír, miszerint az OOo súgószövegét újraírják, az év vége felé megjelenő OOo 1.1-ben ugyan még a régi (zárt, általunk használhatatlan) formában lesz az új tartalom, az új súgórendszer viszont csak a 2004 közepe-vége felé várt 2.0-ban fog bemutatkozni. Addig sajnos úgy tűnik, nem érdemes a súgót fejlesztenünk, mert úgyis a kukába kerül.

A szakmai dolgokon kívül egyebeket is megtapasztaltunk (például a sört méterben is lehet mérni), így a visszautunk egy kisebb késéssel indult, de vasárnap hajnali kettő felé végre hazaértünk.

Verók István
Lassan második anyanyelvének tekinti az angolt és a Javát, így a kezdetektől fordítóként és lektorként segíti az FSF.hu Alapítvány munkáját.

make pingvin && make buli

Nincs az a Linux-felhasználói összejövetel vagy baráti találkozó, amit ne lehetne pingvinszendvicsekkel és sajtigluval feldobni. Szabadjon közzétennünk a fentiek álkódját és egy képet annak bizonyságául, hogy a kód kitűnően működik.

Hozzávalók:
2 doboz krémsajt
1 sajttömb
1 üveg nagy fekete olajbogyó, magozott
1 üveg kis fekete olajbogyó, magozott
1 sárgarépa
1 csomag fogvájó és sárga vagy narancssárga színű sós keksz
1 doboz enyhén sózott füstölt hering (jelképes pingvineledelként, egyébként elhagyható)

```
#!/bin/bash
while Øhes;
do (\
        vÆgd_cs kokra a krØmsajtot_Øs bor tsd_be_vele_a_sajtt mb t;\
        a_sajtcs kokb l_kØsz tsd_el_az_iglu_bejÆrati_alagœtjÆt;\
        fogvÆj val_hœzd_meg_a_h t mb k_hatÆrvonalait;\
        hÆmozd_meg_a_sÆrgarØpÆt;\
        karikÆzd_fel_a_rØpÆt_Ørmevastagsægœra;\
        a_rØpaszeletekbıl_vÆgj_vØkony_Økeket_Øs_tedd_fØlre_csırnek;\
        vÆgd_be_az_ sszes_nagy_olivÆt_Øs_t ltsd_meg_krØmsajttal;\
        a_kis_olajbogy kba_szœrd_bele_a rØpacsırt;\
        szœrd_fel_a_fogvÆj ra_az_olajbogy fejet_Øs_a_nagy_
        ➥olajbogy -testet_Øs_a_rØpacikk-lÆbakat;\
        egy_tÆlon_rendezd_el_a_pingvineket_a_sajtiglu_k r l;\
        a_tÆlon_vagy_a_tÆl_k zelØben_helyezd_el_a_s s_kekszet;\
        tÆlald);
done
```

Az összképet tovább javíthatjuk néhány heringgel. A hering annak a két dolognak az egyike, amivel egy pingvint elégedetté lehet tenni. Elegáns vendégváró étel. A recept fordított fejlesztési eljárással készült, egy fogadáson felszolgált olajbogyó pingvinek és a sajtiglu tanulmányozása nyomán. Reméljük, nincs szabadalmaztatva. Mindenesetre kijelentjük, hogy a recept a saját munkánk eredménye, és a GNU Free Documentation License (szabad dokumentumfelhasználási engedély) feltételei szerint tesszük közzé.

Michael Callaham (pingvinek) és *Jennifer Gentry* (iglu)

Linux Journal 2003. április, 108. szám

Ők mondták

Egy nyereségközpontú programkészítő vállalat nem versenyezhet a nyílt forrás gazdaságosságával – mivel az ingyen van. Továbbá az is kiderült, hogy a nyílt forrás minőségi kipróbálási folyamatával sem kelhet versenyre. Noha a nyílt forrású fejlesztés hosszan elhúzódhat, és a termék általában kevesebb szolgáltatással rendelkezik, mint jogdíjas megfelelője, olyan sokan, olyan gyakran és olyan mértékben próbálják ki, hogy a legtöbb nyílt forrású program állandóbbnak és megbízhatóbbnak bizonyul. Az árupiacon az alacsony ár és a megbízhatóság többet nyom a latban, mint a különféle csecsebecsék.
Christopher Koch, CIO Magazine

Amikor új fájl- vagy nyomtatókiszolgálóra van szükség, ne egy új linuxos gép telepítéséről beszélj, hanem új fájlkiszolgáló vagy nyomtatókiszolgáló telepítéséről. Amíg a megvalósítás kielégítően és megbízhatóan működik, senkit nem érdekel, mi van a háttérben.
Craig Sanders, Debian-fejlesztő és rendszergazda

Arra a felismerésre jutottam, hogy egy hozzáértő Linux-rendszergazda két-háromszor annyi gépet képes ellátni, mint egy Windows-rendszergazda.
Brian Schenkenfelder, n+1

Linux Journal 2003. április, 108. szám

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

A hónap szakmai tanácsai

LILO-tipp vész esetére

Egy korábbi számban szó volt az atombiztos biztonsági mentésről, illetve rendszerindításról. Ehhez a témához szeretnék hozzászólni.
A LILO ismeri a `-R` kapcsolót, ami gyakorlatilag azt jelenti, hogy ezt az új rendszerindítási bejegyzést csak egyszer hajtja végre. A rendszerindítás korai szakaszában telepítheted a saját vészindító üzemmódodat a `lilo -R` paranccsal. A rendszer leállításakor (ha minden jól ment – bár ha biztosra mész, akkor újra befűzöd a `/`-t csak olvasható módba) helyreállíthatod a szokásos rendszerindítási üzemmódot. Így amikor a számítógép váratlanul összeomlik (vagy egy a CTRL-ALT-DELETE billentyűkombináció révén kierőszakolt újraindításkor hirtelen leáll), a rendszer vészindító módban indul el.
Tomas, mlavergne@cfl.rr.com

A gyökérlemezrész rejtélyes ok miatt csak olvasható

Red Hat 7.2-es rendszerem van két merevlemezzel. Mindkettőn három logikai kötet található. Valami történt, ami miatt a gyökérlemezrész írásvédetté vált. Ellenőriztem a könyvtár jogosultságait, de rendben vannak. Amikor viszont állományt próbálok meg létrehozni vagy szerkeszteni a gyökérlemezrész tetszőleges alkönyvtárában, azt a hibaüzenetet kapom, hogy a fájlrendszer csak olvasható.
Jeff Manning, Jeff@VMWorks.com

Győződj meg róla, hogy tényleg a gyökérlemezrészen vagy-e:

```
# cd /
# df .
```

A kimenet *Csatl. pont* oszlopában a / értéknek kell szerepelnie. Ha tényleg csak olvashatóként van befűzve, az a legtöbb esetben fájlrendszerhibát jelent. A hibát még azelőtt ki kell javítani, mielőtt a lemezrészt írható módban fűznéd be:

```
# fsck /dev/hda2
```

A */dev/hda2* értéket helyettesítsd be a `df` kimenetének *Fájlrendszer* oszlopában olvasható értékkel. Végül a gyökérlemezrészt a következő paranccsal fűzzük be írhatóként:

```
# mount / -o remount,rw
```

Christopher Wingert, cwingert@qualcomm.com

Váltás KDM-ről GDM-re

Amikor a számítógépen elindul, és feljön az X, úgy vélem, hogy a KDE bejelentkezéskezelője jelenik meg. Inkább a Gnome GDM-jét szeretném használni, mert többet tud és jobban testreszabható. Hogyan állíthatom be a GDM-et az X alapértelmezett bejelentkezéskezelőjéül?
Gordon Baldwin, gordonbaldwin@bigpond.com

Jelentkezz be rendszergazdaként, és a */etc/sysconfig/desktop* állományban írd át a

```
DESKTOP="KDE"
```

sort erre:

```
DESKTOP="GNOME"
```

Keith Trollope, ktrollope@san.rr.com

Kapcsolódás a Microsoft-hálózathoz

Hogyan lehet a linuxos számítógépet windowsos egyenrangú hálózatba illeszteni? Van egy windowsos egyenrangú hálózatom, amelyben egy Windows XP és három vagy négy Windows 98 csatlakozik egymáshoz. Ezek a számítógépek állandó IP-címmel rendelkeznek a 10.1.1.1–10.1.1.14 tartományban. Minden számítógép teljesen meg van osztva, nincsenek jelszavak. Legtöbbször én vagyok az egyetlen felhasználó. Hogyan vehetném rá a Windowsokat, hogy lássák a Linux meghajtóit, és viszont?
J. G. Owen, owen_labs@worldnet.att.net

A Samba segíthet a cél elérésében. A Samba telepítése zavarosnak tűnhet, mert a leírás nem teszi világossá az első ponttól kezdve, hogy a feladat gyakorlatilag két részre bomlik. Windows környezetben az ügyfél és a kiszolgáló egybeolvad. Ha a hálózat működik, tallózhatunk más rendszereket, és azok is tallózhatják a miénket. A Sambánál mindegyik oldal elérheti a másikat.
A Samba telepítése után nem kell sokat kínlódni ahhoz, hogy a többi rendszert elérhessük. Az `smbmount` paranccsal lehet a távoli rendszerek megosztásait befűzni. Ha telepítve van valamilyen névfeloldás (pl.: DNS), akkor az `nmbd`-t és az `smbd`-t általában nem kell futtatni. Az állandó megosztásokat felvehetjük a */etc/fstab* állományba, az alábbihoz hasonlóan:

```
//machname/share  /mountpt smbfs
➥username=xxx,password=yyy  1 1
```

Ha azt szeretnénk, hogy a linuxos gépet más gépekről el lehessen érni, akkor kiszolgálóként kell beállítani. Ezt az *smb.conf* állomány szerkesztésével tehetjük meg. Itt kell megadni a kiszolgáló tulajdonságait és a létrehozandó megosztásokat. Ezek után futtassuk az `nmbd` és `smbd` démonokat, amelyek megvalósítják ezt a szolgáltatást. Ha el akarjuk kerülni az elsődleges tartományvezérlővel kapcsolatos, néha zavaros munkát, olyan módon fordítsuk le a Sambát, hogy támogassa a rendszerünkön bevezetett szokásos jelszószolgáltatásokat. Így a felhasználókat a szokásos módon adhatjuk hozzá a rendszerhez, és az `smbpasswd` segédprogrammal létrehozhatjuk a fájlt, amelyet a Samba valójában használ. Ez viszonylag sok kézi munkát igényel (a ➲ http://org.samba.www címen az „Unofficial HOWTO" írja le), de a célnak megfelel. Ha a megosztásokat tallózhatóvá szeretnénk tenni, akkor hozzá kell adni a guest vagy a no-name felhasználói fiókot is.
Chad Robinson, crobinson@rfgonline.com

A Windows különféle változatai eléggé érzékenyek a hálózati környezet beállításaira. A leghelyesebb a */etc/samba/smb.conf* állományt szerkeszteni. A `workgroup =` sorba a munkacsoport nevét kell írni. A `netbios name =` sorban adjuk meg a számítógépünk nevét. Ha úgy telepítjük, a jelszavakat a program a */etc/password* állományból veszi.
Christopher Wingert, cwingert@qualcomm.com

Új termékek

Appro 2U kiszolgálócsalád

Két új kiszolgálócsaláddal jelentkezett az Appro Computers. Az egyik Xeon processzort, a másik Athlon processzort használ, és mindkettő 2U magas, keretbe szerelhető dobozban kapott helyett. A 2224X két Xeon processzort, négy SCSI- vagy IDE-merevlemezt, legfeljebb

12 GB DDR-es ECC memóriát (PC2100) és két 10/100-as ethernetkaput tartalmaz. A 2228X rendszer ugyanilyen, csak abban nyolc SCSI-meghajtó van. Az athlonos gépek két Athlon MP processzort, legfeljebb 4 GB DDR-es ECC memóriát (PC2100) és két 10/100-as ethernetkaput tartalmaznak.
A 2224-es kiszolgáló négy SCSI vagy IDE-merevlemezt tartalmaz, a 2228-as pedig nyolc SCSI-merevlemezt.
Adatok: Appro International, 446 South Abbott Avenue, Milpitas, California 95035,
telefon: 800-927-5464,
➲ http://www.appro.com

SCOoffice Server

Az SCO Group SCOoffice Server programja kis- és közepes vállalkozások számára készített irodai kiszolgálócsomag, amely a következő összetevőkből áll: SCO Linux 4.0, SCOoffice Mail Server 2.0 és SCOoffice Base Server. A SCO Linux 4.0 a UnitedLinux 1.0-s terjesztésen alapul. A Mail Server képes több levelezőprogram, levelező-, naptár- és címjegyzék-szolgáltatásainak kiszolgálására. A Base Server segítségével grafikus felületen keresztül lehet beállítani a nyomtatókat, a fájlmegosztásokat, a hálózatot, de más megosztott erőforrások és segédeszközök is vannak benne.
Adatok: The SCO Group, 355 South 520 West, Suite 100, Lindon, Utah 84042,
telefon: 801-765-4999,
➲ http://www.sco.com

MontaVista CEE

A MontaVista Linux Consumer Electronics Edition (CEE) beágyazott operációs rendszer és fejlesztőkörnyezet szórakoztatóelektronikai eszközökhöz. Jellemzői: dinamikus energiaellátás-kezelés, bővített állományrendszer, fejlesztőeszközök a rendszer teljesítményének hangolásához, a processzor és a külső eszközök támogatása, fejlesztőeszközök rendszer- és alkalmazásfejlesztéshez, valamint telepíthető segédeszközök, programkönyvtárak, eszközmeghajtók és más futásidejű összetevők százai. A CEE binárisként és forrásban is elérhető, alkalmazási területei lehetnek a valós idejű, többfeladatú és többszálú működést igénylő eszközök, például a mobiltelefonok, a set-top boxok, a digitális televíziók és az autóba szerelhető távközlési eszközök.
Adatok: MontaVista Software, 1237 East Arques Avenue, Sunnyvale, California 94085,
telefon: 408-328-9200,
➲ http://www.mvista.com

Astaro Security Linux v4

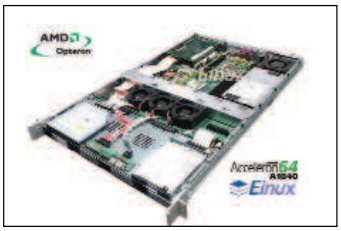
Az Astaro Corporation a New York-i Linux World kiállításon jelentette be az Astaro Security Linux 4-es változatának megjelenését, ami egy internetes biztonsági program. Az Astaro Security egy csomagban tartalmaz tűzfalat és VPN-átjárót, levélszemétszűrőt, tartalomszűrőt, címletiltást és vírusvédelmet. Az Astaro Security v4 újdonságai között említhetjük meg a VLAN és a WLAN támogatását, a POP3-as postafiókok vírusvédelmét, a heurisztikus levélszemétszűrést, valamint a Radius és LDAP protokollok jobb támogatását a helyi és távoli felhasználók azonosítására.
A VPN IPSec és előre kiosztott kulcsok segítségével végzi a hitelesítést, a PPTP pedig 128-bites titkosítást nyújt. A VPN átviteli sebessége 115 Mb/s, a tűzfal átviteli sebessége 735 Mb/s, mindez egy 1266 MHz-es processzorral.
Adatok: Astaro Corporation, 67 South Bedford Street, Suite 400W, Burlington, Massachusetts 01803,
telefon: 781-229-5880,
e-mail: info@astaro.com,
➲ http://www.astaro.com

Umigumi

Az OpenBrick Community új projektet indított, amelynek az Umigumi nevet adta. A cél az operációs rendszerek telepítésének és terjesztésének egyszerűsítése flashkártyák használata révén. A felhasználó által adott bemenő adatoknak megfelelően az Umigumi rendszerindító kóddal ellátott flashkártyát hoz létre. A kártya azután egy másik rendszerbe illeszthető, amiből útválasztó, tűzfal, virtuális magánhálózat, OGG-lejátszó, nyomtatókiszolgáló, vékony ügyfél stb. is válhat. Az Umigumit eredetileg az OpenBrickhez tervezték, de bármilyen beágyazott felületen működőképes, például PowerPC, StrongARM/XScale, Mips/Mipsel, SuperH, SPARC és 68 K, amin nyílt forrású operációs rendszer fut.
➲ http://www.umigumi.org

Acceleron64 Opteron kiszolgálók

Az Einux bejelentette új, két 64-bites AMD Opteron processzorra épülő, keretbe szerelhető kiszolgálócsaládját, az Acceleron64 sorozatot. A család első két tagja az A1840-es nagyvállalati szintű kiszolgáló és az A1820-as alacsonyabb szintű kiszolgáló. Az A1840 1U magas kiszolgáló négy menet közben cserélhető merevlemez beépítési lehetőségével. A RAID 5 hozzáadható, valamint rostcsatornán és iSCSI-n keresztül SAN-okhoz kapcsolható. Az A1820 1U magas kiszolgáló két menet közben cserélhető 350 W-os hibatűrő tápegységgel. Mindkét kiszolgálóba legfeljebb 12 GB DDR333 ECC memória, egy Ultra 320 SCSI-vezérlő, két gigabites ethernetcsatoló és menet közben cserélhető ventilátor szerelhető.



Adatok: Einux, 1313 North Milpitas Boulevard #110, Milpitas, California 95035,
telefon: 866-883-4689,
➲ http://www.einux.com/about/acceleron64.php

Linux Journal 2003, 108. szám

A szubkontinens csattanója

A Tux és a Bux közti világméretű háborúban hol van a legjobb hely, ahol a pingvinre fogadhatunk? Próbáljuk meg Indiát!

A William és Melinda Gates Alapítvány 2002. november 11-én százmillió dolláros kötelezettségvállalást hozott nyilvánosságra a HIV/AIDS-megelőzésre Indiában.
A bejelentésre ***William Gates*** agyonreklámozott indiai látogatását megelőzően került sor. Az eseményre időzítve a Microsoft bejelentette, hogy üzleti kapcsolatait Indiával egy megközelítőleg négyszázmillió dolláros befektetés keretében kívánja elmélyíteni.
A Windows XP-t úgy tervezték meg – az operációs rendszer előző változataitól eltérően –, hogy ne működhessen anélkül, hogy a felhasználó ne hívná fel vagy jelentkezne be a Microsoft hitelesítési eljárásába, ahol termékét működőképessé teszik. Ha a Microsoftnak lehetősége lesz rá, Indiában minden egyes Windows-példányért fizetni kell majd.
A Windows XP nem olcsó, de még így is képes versenyre kelni egy ingyenes operációs rendszerrel. Gates kijelentette, a Microsoft nem tervezi, hogy operációs rendszerére a szokásosnál nagyobb kedvezményt adjon az oktatási intézmények számára.
Az adott versenyhelyzetben hogyan győzheti le a Microsoft a Linuxot egy olyan gazdaságban, ahol az MS-termékek luxusárunak számítanak?
Nem tudom, és nem is nagyon érdekel – ez legyen a Microsoft gondja. Ami viszont érdekes számomra, és emiatt hoztam fel a Microsoft problémáját, az egy másik ügy, mégpedig az, hogy az indiai fejlesztők vezető szerephez juthatnak a Linux-mozgalomban.
Van néhány nagyon becsvágyó Linux-fejlesztő Indiában. Vegyük például ***Rajesh Jain***-t. Miután a Linux Journal oldalán írtam Indiáról, egy olvasó felhívta a figyelmemet arra, hogy mit szeretne Jain új vállalkozásával, az Emergickel elérni: Jain ***Michael Robertson***-nal egyetemben szintén az internetes kiadói üzletben volt érdekelt, az IndiaWorld nevű cégnél, amit később eladott. Úgy gondolom, ő ama emberek egyike a Lindows mellett, akik kezdeni szeretnének valamit a Linuxszal, csak ezt más módon kívánják megtenni – és ennek határozottan óriási piaca van az alsó piaci szegmensben, ha Jain a megfelelő lépéseket teszi meg (bármik is legyenek azok).
Jain így vázolja fel terveit: az Emergic egy olyan program kifejlesztésén fáradozik, ami a technológiai költségeket a tizedére csökkentve azt elérhetővé teszi a világ fejlődő piacain tevékenykedő felhasználók és vállalatok számára.
Az Emergic kezdi megvalósítani ***Bill Gates*** látomását: „tegyünk egy-egy számítógépet minden íróasztalra és otthonba”. Ez a látomás nem volt képes túllépni a világ tízezer legnagyobb vállalatán és ötszázmillió felhasználóján, akiknek túlnyomó többsége a világ fejlett részén él.
Az Emergic a következő ötszázmillió ember és 25 millió kis- és középvállalkozás számítástechnikai felülete kíván lenni. Ők eddig azért nem voltak képesek alkalmazni a technológiát, mert méregdrága volt.
Az Emergic a világ fejlődő országainak lakosait célozza meg. Azokat, akikhez még nem gyűrűzött be mélyen a technológia, és akik számára talán ez az utolsó alkalom, hogy bekapcsolódjanak a világ értékláncába, javítva ezáltal az ott élő emberek életszínvonalát.
Jain 125–150 dollár közé teszi vagy még lejjebb csökkenti a gép árát, a programét pedig havi 5–10 dollárra. Mindezt úgy kívánja elérni, hogy az elavult számítógépeket vékony ügyfélként használja az alkalmazások futtatásához, a szokásos linuxos forgatókönyv szerint.
Jain a széles sávú kapcsolatban bízik, és elsősorban a vállalati körben való felhasználásról beszél: a linuxos kiszolgálóalapú számítástechnika immár lehetségessé vált, mivel a helyi hálózat sebessége elérte a 100 Mb/s-t, ami sokkal több adat átvitelét teszi lehetővé ugyanazon hálózaton keresztül. Az eredmény egy „vastag kiszolgáló” lett, alapvetően egy új asztali számítógép 1 GB operatív tárral és két merevlemezzel, programból megvalósított RAID-dal, ami könnyedén ki képes 30–40 felhasználót szolgálni. Egy ilyen kiszolgáló kb. 1500–2000 dollárba kerül, egy főre lebontva az ügyféloldali költség nem éri el az 50 dollárt. Mindezeket összeszámolva a vékony ügyfél és vastag kiszolgáló kombináció nemcsak az eszköz- és programköltségeket csökkenti 90 százalékkal, de az informatikai vezető számára teljes felügyeletet jelent az ügyfelek felett.
Jainnek rengeteg további terve van, de ahelyett, hogy elmélyednénk bennük, nézzünk meg inkább néhány más folyamatot, amelyek magában a kultúrában mennek végbe. Vegyük például az indiai Ahmedabadi Vezetőképző Intézetben zajló munkát. ***Peter Day*** a BBC Newsban a következőket írja: Az indiai Vezetőképző Intézet nem pusztán a jövendőbeli internetes milliárdosokért és azokért a diákokért létezik, akik új kék köpenyt hordanak. Nem miattuk érkeztem ide. Azért jöttem Ahmedabadba, hogy együtt ebédeljek a legtalálékonyabb emberekkel, akikkel valaha is találkoztam – feltalálókkal és ezermesterekkel, akik a vidéki India földjeiről és falvaiból érkeztek, ahol az egymilliárdos lakosságból több mint hétszázmillióan még mindig itt élnek. Kezükkel ették a rizst, a dhált és a zöldségeket, miközben lelkesen beszéltek a találmányaikról és az ötleteikről.
Ezek az emberek a Kezdeményezés a kutatásért, a fenntartható technológiákért és az innovációért egyesület (Society for Research and Initiatives for Sustainable Technologies and Innovation) égisze alatt gyűltek össze. A szervezet rövidítése SRISTI, amelynek szankszrit jelentése: alkotni.
Összegezve: 1. rendkívül talpraesett emberekről van szó, akik nagyon sokan vannak; 2. jelen esetben is a Linux, valamint a nyílt és a szabad forrású programokhoz hasonlatos értékekről beszélünk. Ezért nem lennék meglepve, ha először itt, Indiában győzné le a Tux Buxot.

Linux Journal 2003. április, 108. szám

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

Közösségi programok

A közösségi fejlesztésű program és a közösségi fejlesztésű szabvány ugyanannak az éremnek a két oldala. A szabványok abban segítenek, hogy a szabad újítás – a nyílt forráskód és a Linux lényege – ne legyen az akadálya olyan programok írásának, amelyek hatékonyan működnek együtt. – véli ***Brian Behlendorf***, az Apache Group társalapítója, a CollabNet főmérnöke.

Még 1997-ben linuxos programfejlesztők egy csoportja azon tűnődött, mit lehetne tenni a Linux-terjesztések közötti kicsiny, de bosszantó eltérések megkerülése érdekében. Mi több, az egyes terjesztések különböző változatai közötti eltérésekkel is meg kellett küzdeniük.

A csoport nyílt forráskódú programfejlesztői számára a kérdés az volt, hogyan tölthetnének több időt új szolgáltatások és javítások megvalósításával ahelyett, hogy órákon át azt ellenőriznék, hogy a programjuk mindegyik Linux-változattal működőképes-e.

A nem szabad programok fejlesztői ugyanezzel a kérdéssel szembesültek, de nekik emellett fizető ügyfeleik bizalmára és alkalmazottaikra is gondot kellett fordítaniuk. Valamit tenni kellett.

Szerencsére szinte mindenki egy véleményen volt, a legnevesebb fejlesztőktől kezdve a Linux-terjesztések készítőin át a felhasználókig. Hamarosan megalakult a Linux Standard Base (LSB) kezdeményezés, aminek ***Alan Cox*** tervezett honlapot, ***Bruce Peters*** vállalta a vezető szerepét, ***Jon „maddog” Hall*** pedig tanácsadóként segített. Minden nagyszerűen alakult, ***Linus Torvalds*** is kiállt az erőfeszítések mellett, ám a csoport nem mérte fel teljesen, hogy milyen hatalmas feladatra vállalkozott. Nem csupán egy olyan szabvány megalkotására volt szükség, ami megfelel a fejlesztők, a terjesztéskészítők, a vállalkozások és a felhasználók igényeinek, de valóban működőképessé kellett tenniük, mégpedig első nekifutásra.

Tekerjük előre a szalagot a kétezres évig. Az LSB a 0.02 változatnál tartott, és a fejlesztők egyik csoportja tanácsért fordult az LSB-csoporthoz a Linux-honosításokat leíró szabvány kifejlesztésével kapcsolatban. Néhány megbeszélés után nyilvánvaló lett, hogy új keretre van szükség, ami lehetővé teszi, hogy mindkét munkába többen kapcsolódhassanak be, miközben a két tervezet független és közösségi alapú marad. Ez volt a Free Standards Group születésének pillanata.

A Free Standards Group egy olyan kaliforniai nonprofit vállalkozás, ami a nyílt szabványok fejlesztésének, alkalmazásának és népszerűsítésének segítségével arra törekszik, hogy felgyorsítsa a nyílt forrású megoldások terjedését és elfogadottságuk növekedését.

Nem sokkal 2000 második fele, vagyis a megalakulása után a Free Standards Group új energiákat hozott működésbe mind a nyílt kódú programok fejlesztői, mind az informatikai ipar szereplői között. Az olyan fejlesztők, mint ***Ted Ts'o***, ***Stuart Anderson*** és ***Dan Quinlan***, és olyan cégek, mint a Red Hat, a SuSE, a HP és az IBM egyaránt kivették részüket a munkából.

Ez a munka nem csupán egy szabvány rögzítését jelentette: a Linux-rendszer egységesített, teljes körű működési leírását készítették el, valamint olyan eljárásokat, amelyekkel a szabványos rendszerre fejleszteni, illetve azt ellenőrizni lehet. Az LSB például tesztkészleteket tartalmaz az operációs rendszerhez, a felhasználói programokhoz és a fordítási környezethez egyaránt. Ezenkívül magában foglalja a fordítási környezetet, a példamegvalósítást, az alkalmazáscsomagot és a teljes körű leírást.

- ***Szabványleírás***: az LSB-megfelelő operációs rendszerek működését írja le. Nem azt adja meg, hogy a rendszermag, rendszerkönyvtárak vagy más központi elemek melyik változatát kell alkalmazni, kizárólag az egyes összetevők viselkedését szabja meg. Ez lehetővé teszi, hogy a fejlesztők csak az API-val és az operációs rendszer API-val legyenek kapcsolatban.
- ***Tesztkészletek***: az operációs rendszer, az alkalmazások és a fordítási környezet kipróbálását teszik lehetővé.
- ***Fordítási környezet***: elszigetelt környezet, amibe a fejlesztők `chroot` segítségével lépnek be, hogy LSB-megfelelő alkalmazásokat fordítsanak.
- ***Példamegvalósítás***: elszigetelt környezet, amibe a fejlesztők `chroot` segítségével lépnek be, hogy az LSB-megfelelő alkalmazások próbafuttatását elvégezzék.
- ***Alkalmazáscsomag***: nyílt forráskódú alkalmazások gyűjteménye, amik az LSB-megfelelő operációs rendszerek terheléses tesztelésére használhatók.

Körülbelül fél évvel a teljes LSB kiadása után indult útjára az LSB-tanúsítvány. A tanúsítvány mind a Linux-terjesztők, mind a -fejlesztők számára lehetővé tette annak bizonyítását és bemutatását, hogy termékeik megfelelnek a szabványnak. Az LSB-tanúsítvány bevezetését követő hat héten belül a Mandrake, a Red Hat és a SuSE is megpályázta a tanúsítványt, és sikeresen meg is szerezte.

Mára minden nagyobb Linux-terjesztő megszerezte az LSB-tanúsítványt.

A Linux-terjesztések közötti eltérésekkel kapcsolatos vita nyugvópontra jutott. A programfejlesztők biztosak lehetnek benne, hogy LSB-megfelelő rendszerekre fejlesztett programjuk minden egyes LSB tanúsítvánnyal rendelkező rendszeren változtatás nélkül futtatható lesz. A felhasználók számára is előnyt jelent a terjesztések közötti átjárhatóság és a felhasználói programok nagyobb száma.

Szabványaik nagy sikerű bevezetése ellenére a Free Standards Group és annak LSB- és Openi18n-csoportjai nem ülnek ölbetett kézzel. Tovább haladunk meglévő szabványaink kiterjesztése felé és az olyan új területekre, mint a nyomtatás és a felhasználói felületek szabványai.

Amennyiben érdekel a Linux jövője, csatlakozhatsz hozzánk. Részt vehet a munkában minden egyén, civil szervezet, oktatási intézmény, cég vagy kormányzati szerv. További tájékoztatás a ➲ http://www.freestandards.org honlapon található.

Linux Journal 2003. április, 108. szám

Scott McNeil,
A Free Standards Group egyik alapítója és ügyvezető igazgatója.

A nagy egyesített munkafelület

Hogyan vehetik rá a fejlesztők és a terjesztések a különféle programokat, hogy együtt jobban működjenek?

A Linux Journal weblapján 2002 decemberében megjelent egy cikkem (➲ http://www.linuxjournal.com/article/6476) a Red Hat 8.0 új Bluecurve nevű munkafelületéről, ami a Gnome és a KDE rendszerből egyaránt tartalmaz programokat. Számos visszajelzést kaptam, sőt néhány javaslat is érkezett, hogyan lehetne olyan alkalmazásokat írni, amelyek képesek együttműködni más eszközkészletekkel írt alkalmazásokkal, és minden munkafelületen jól működnének.

A kérdés

Az összevont munkafelület olyan szerkezet, aminek az elemei együttműködnek egymással, és soha nem kényszeríti arra a végfelhasználót, hogy ugyanazt a dolgot többször ismételgesse. A jelenlegi alapértelmezett megoldások (például a Gnome és a KDE) használata során gyakran ütközünk korlátokba, ha keverni szeretnénk az összetevőket.

A grafikus ügyfelek közötti valamennyi kapcsolat – az egyszerű húzd és ejtsd megoldástól kezdve az ikonok, menük, címek és más metaadatok kezeléséig, illetve általában az ablakszintű kapcsolattartásig – csak az ügyfelek és az ablakkezelők egy bizonyos alhalmazán működik kifogástalanul. Ugyanez vonatkozik a központilag kezelt és jól kinéző betűtípusokra.

No és mi a helyzet a nemzeti sajátságok kezelésével? Hány program létezik még, amit nem angol nyelven vagy billentyűzettel kínszenvedés használni?

A KDE és a Gnome két olyan alkalmazásgyűjtemény, ami úgy tűnik, sokat megold a felsorolt nehézségek közül. De valóban az eszközkészlet-központú, mindent vagy semmit alapú megközelítést lehet a legjobban fenntartani? Mi lesz, ha valaki még jobb eszközkészlettel áll elő? Mi történik a más eszközkészletekben írt alkalmazásokkal vagy az egész „best of breed” (a legjobb fajta) eszmeiséggel, ami olyan természetes a szabadprogram-világ felhasználói számára?

Más szavakkal: mi a legjobb módja annak a bizonyos legjobb alkalmazás a fejlesztésének, amit a kedvenc munkafelületünkhöz, mások kedvencéhez, illetve a jövő munkafelületeihez használunk?

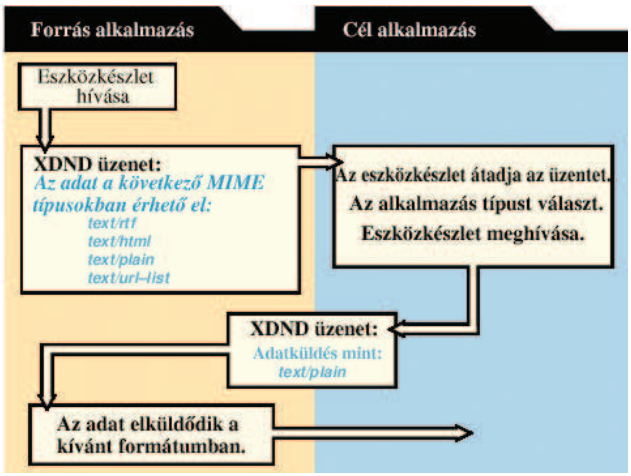
Megoldások

A kerék újrafeltalálása bolondságnak tűnhet, de sokkal kevésbé balgaság, mint olyan kereket fejleszteni, aminek újfajta utakra van szüksége. Amennyiben tehát létezik már húsz csevegőügyfél Linux alá, de úri kedvünk úgy tartja, készítsünk egy újabbat; azonban ne feledkezzünk meg arról sem, hogy együtt kell tudni működnie a már meglévőkkel.

Kezdjük józanul, és aknázzuk ki a helyes szabad szabványokat és protokollokat! Ezután ellenőrizzük, hogy a kiválasztott könyvtár vagy eszközkészlet szintén támogatja-e őket. Sok fejlesztő már rájött, hogy hosszú távon ez a nyerő megközelítés. ***Havoc Pennington***, Gnome-fejlesztő ezt írja: „Az alkalmazás és a futásidejű környezet közötti kapcsolattartásnak mindenképpen dokumentált eszközkészlet-független protokollokon és fájlformátumokon keresztül kell zajlania.” Lássuk, hogyan is fest mindez a gyakorlatban!

Fájlformátumok

A hivatali formátumok, de a strukturált adattárolás számos más területén is talán az egyik legígéretesebb dolog az Oasis csoport munkája. Az Oasis fájlformátumai az OpenOffice.org formátumaiból fejlődtek ki, így könnyebb idők következnek az



1. ábra XDND, az X húzd és ejtsd protokollja

OpenOffice.org fejlesztői számára. De minden program, az Emacstól a Koffice-on keresztül a jövő szövegszerkesztőjéig fel tudja használni.

Ezt az utat követve nincs szükség újabb könyvjelzőformátum készítésére. Az XBEL (XML Bookmark Exchange Language) olyan könyvjelző-csereformátum, amelyet a Galeon és a Konqueror is régebb óta használ.

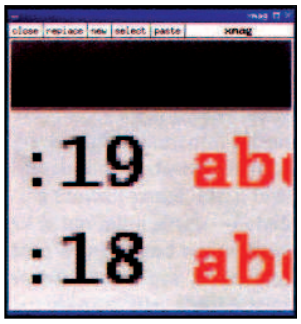
A csereformátum ötlete valamennyi beállításfájlra vonatkozik. A Unix és a Linux még mindig az eszközkészlet-megközelítést alkalmazza: nincsenek mamutok, csak sok kis darabka, amelyek kiválóan végzik a saját feladatukat. Ennek az az egyik hasznos következménye, hogy valamennyi azonos feladatot ellátó alkalmazás ugyanazt vagy egymáshoz nagyon hasonló értékkészletet ismer és állít be.

A csevegőügyfeles példánknál maradva: nem lenne túl okos külön beállításfájlokat készíteni Qt_chat_app, GTK_chat_app, és MyToolKit_chat_app és egyéb neveken a különféle eszközkészletekhez. Alapvető különbség van a fájl és annak előállítása, frissítése között. Használhatunk ugyan különféle szerkesztési módszereket (vi, vezérlőpult stb.), de hosszú távon csak az számít, hogy az összes létező csevegőügyfelünk egyetlen ***~/.chatrc*** fájlt használjon, és ne panaszkodjon, ha a beállításokat egy másik ügyfél megváltoztatja. A Usenet hírolvasó

például már régóta a szabványos ***.newsrc*** fájlt használja. Amíg az általunk készítendő alkalmazásfajtához még nem alakult ki szabvány, elfogadható egyezségnek tekinthető, ha az új programunk induláskor más hasonló ügyfelek beállításait tölti be, és azokat használja alapértelmezettként.
A MIME-típust fájlnév alapján a MIME-adatbázis végigkeresésével meg lehet határozni. Ezt szintén egységesíteni lehet, így minden munkafelület és program garantáltan azonos adatokat olvas be; ennek megvalósítási lehetőségeiről a
➲ http://freedesktop.org lapon vázlatos leírást találunk.

Grafikus felületek

Sokak szerint az X megfogja mutatni érettségét – és ilyenkor olyan ambiciózus, messzire mutató projektekre céloznak, mint a Fresco. Anélkül, hogy ilyen messzire mennénk, elmondhatjuk, sok mindent tehetünk annak érdekében, hogy az ügyfelek és az ablakkezelők bármilyen kombinációban helyesen együttműködjenek. Az ilyenfajta nehézségek megközelítéséhez kiindulási alapként a
➲ http://freedesktop.org honlapot javaslom, különös tekintettel a szabványok fejezetre. Emlékszünk még, mit is mondtunk? Az azonos típusú alkalmazásoknak azonos beállítási értékkészleten kell osztozniuk. A magszolgáltatásokra hivatkoztunk, de ugyanez a megközelítés igaz minden GUI-szintű beállításban az asztal összes eleme esetén. Az olyan jellemzőket, mint a háttérszín, a duplakattintás időzítése, minden program használja, mégis minden felhasználó csak egyszer és azonosan állítja be, függetlenül attól, hogy milyen eszközkészlet-keveréket használ éppen.



2. kép Élsimított szöveg

3. kép Szöveg hagyományos Xterm alatt

Az X Resource Managert nem készítették fel felhasználói beavatkozást igénylő kezelésre, s mivel a ***~/.Xdefaults*** tartalmát egyesíti más forrásokból származó adatokkal, nem éppen eszményi háttér GUI-beállítási eszköznek. Inkább tekintsük meg az Xsetting-meghatározásokat, amiket ennek és más nehézségeknek a legyőzésére készítettek.
Lépjünk tovább! Mi következik, miután az egyes alkalmazásosztályokat, a megjelenésüket, illetve a helyi felhasználók cselekedeteire adott válaszaikat beállítottuk? A más alkalmazásokkal és az ablakkezelővel történő grafikus kapcsolattartás kérdése. Már most is lehetőségünk van szövegeket húzni át egyik ablakból a másikba, legalábbis elméletileg, amennyiben az összes résztvevő követi az XDND protokollt. Jelenleg nemcsak a GTK+ és a Qt, de még a parancsfájlközpontú eszközkészletek is támogatják, mint a Tk és a Perl/Tk, illetve az olyan viszonylag kívülállók, mint a FOX.
Az *1. ábra* szemlélteti, hogyan működik az XDND. Amikor az alkalmazásnak valamit át kell küldenie egy másiknak, kiadja a megfelelő eszközkészlethívást. Az eszközkészlet az XDND protokollon keresztül mindent elküld a másik eszközkészletnek, majd a válasz ugyanezen az úton kerül vissza a programhoz.
A forrás az eszközkészleténekmegadja az összes általa támogatott MIME-típust, amik lehetnek különféle szöveges formátumok, képek vagy fájlnevek. A célnak meg kell mondania, hogy melyik formátumot ismerte fel az adott listából. Így a formázott szöveget két szövegszerkesztő között is átadhatjuk, anélkül, hogy a karakterkészletet, a színeket és a többi formázást elvesztenénk. Ugyanakkor ha egy hasábot átemelünk mondjuk a KOffice-ból a vi alá, akkor a formázás elvész, de megmarad a tartalom.
Amikor a húzd és ejtsd nem működik, a hiba oka csaknem mindig az alkalmazás XS-ében keresendő, ami nem adta meg az összes MIME-típust, csak saját formákat alkalmazott, vagy hibásan használja az eszközkészletet, esetleg a protokollt. Ez gyakorlatilag azt jelenti, hogy a hibajelentést az alkalmazáshoz kell benyújtanunk, és nem az eszközkészlethez vagy az XDND-hez.
Az X-ügyfelek közti, illetve az ügyfelek és az ablakkezelő közötti egyéb kapcsolatoknak az ICCCM szabványt (Inter-Client Communication Conventions Manual), illetve az Extended Window Manager Hintset, röviden EWMH-t, a korábban NetWM néven ismert leírást kell követniük. Megjegyezzük, hogy az ICCCM még a jó öreg X vágólap részletes ismertetését is tartalmazza. Kezelésükről részletes magyarázatot kaphatunk ***Keith Packard***-tól és ***Jamie Zawinski***-től.
A második ablakkezelési szabvány az ICCCM-re épül. Az összes olyan ablakkezelési képességet kezeli, amit az elődje nem tartalmazott, mivel azok az ICCCM létrehozása után jelentek meg. Az EWMH eredetileg az akkori Gnome ablakkezelő leírás helyettesítésére készült, de ma már (helyesen) bármilyen munkafelületen megvalósítható és támogatott. A Gnome 2 és a KDE 3 egyaránt támogatja az EWMH-t, így minden alkalmazás, ami használja, várhatóan mindkét környezetben, illetve azok elemeivel szépen fut majd. Azoknak a fejlesztőknek, akik az EWMH-ról ennél a gyors áttekintésnél pontosabb leírásra vágynak, érdemes belenézniük a KDE ***netwm.h*** állományába.

Betűtípusok és kódolás

Mindent elmondtunk már, ami a nagy egyesített munkafelülethez szükséges lehet? Nyilvánvalóan nem. Gondoljunk csak a Red Hat 8.0-ra. Szinte mindenki, még azok is, akiket a hideg ráz ki a Gnome/KDE keverésétől, egyetértenek abban, hogy legalább a betűk sokkal jobban néznek ki. Mi is könnyen becsempészhetjük a saját terjesztésünkbe ezt a varázslatot.
Először is, a fejlődés az élsimítás érdeme, ami szebbé varázsolja a karaktereket, pixeleket szúrva be a megfelelő helyekre. A *2.* és *3. kép* ugyanazt a szöveget mutatja be egy szabványos `xterm` és egy élsimításos változatban.
Másodszor a Red Hat 8.0 az első vezető terjesztés, ami a lehető legtöbb UTF-8 kódolást használta (erről még később lesz szó), illetve beépítette az `xft2` és ***fontconfig*** könyvtárak által nyújtott ügyféloldali betűtípus-kezelő rendszert.
A ***fontconfig*** képes kitalálni, hogy milyen betűtípusok érhetők el a rendszeren és hogy milyen dokumentumhoz melyik felel meg a leginkább. Ha ez megvan, az `xft2` mondja meg az X-kiszolgálónak, hogy mit kell kirajzolnia. Mindkét könyvtárnak kapcsolatban kell állnia a FreeType vagy egy azzal egyenértékű raszterizáló motorral. Ez a következőket jelenti:

- Mostantól nincs szükség betűtípus-kiszolgálókra.
- Az új betűtípusok telepítése (akár rendszergazdai jelszó nélkül is) sokkal egyszerűbb lett.
- Minden alkalmazás, ami a ***fontconfigon*** alapul, az összes betűtípus-beállítást egyazon XML-fájl(ok)ból olvassa ki, amit bármilyen felülettel szerkeszthetünk.

4. kép Emacs Red Hat 8.0 alatt

- A betűtípus-kezelés mostantól (bármely alkalmazásban) e két könyvtár sebességével történik, és nem magának az X-nek a sebességével.
- Mivel a ***fontconfig***-nak nincs szüksége se `xft2`-re, sem más X-szel kapcsolatos elemre, bármibe beágyazható, ami betűtípusokat kezel – akár nyomtatómeghajtókba vagy könyvtárakba is. A ***libgnomeprint22*** pontosan ezt is teszi.

Az xft2/fontconfig rendszer másik ügyes megoldása, hogy nem mindent vagy semmit alapú. Békésen megfér a hagyományos betűtípus-kiszolgálók mellett, amik a régebbi alkalmazásokhoz esetleg szükségesek lehetnek. Ez történt a Red Hat 8.0 esetében is. Az `xft2` pedig az új és a régi XFree86-kiszolgálókkal is beszélgetni tud.

Előfordul azonban, hogy a szép szimbólumok rajzolgatása még nem elegendő, és az alkalmazásunknak fejlett szövegmegjelenítésre is szüksége van. Ilyenkor kerül a képbe a ***fontconfig***-on alapuló Pango program. A Pango szintén a többé-kevésbé monolitikus Gnome munkafelületen született, de jelenleg már bármilyen környezetben futtathatóra tervezik.

Egy utolsó gondolat a betűtípusokról: a szebb rajzolat jó dolog, de ha kizárólag számjegyekhez és az angol ábécéhez használhatjuk őket, akkor az egész nem igazán érte meg, nemde?

Ha a kelet-európai, afrikai vagy ázsiai nyelveket nézzük, az ASCII még a „Szia, Világ!” megfelelőjét sem tudja visszaadni. A fejlesztő oldaláról nézve ez annyit jelent, hogy az alkalmazásokat az első naptól kezdve többnyelvűre kell tervezni, és minden, már létező programot ellenőrizni kell, hogy biztosan működik-e. Ez nem csupán egyszerű javaslat. Ha valaki azt hiszi, hogy biztonságban van, mert csak ASCII-t használ és csak néhány parancsfájlt alkotott, az gondolja végig még egyszer a dolgot. Amikor a legközelebb fejlesztünk, és a kódunk egy nemzetköziesített parancsértelmezőben, terminálban vagy ablakkezelőben indul majd el, könnyen lerobbanhat. Perlben íródott, véletlen aláírásokhoz készült programom pontosan emiatt állt le Red Hat 8.0 alatt, és csaknem három hónapig három különböző listán senki sem tudott használható megoldást javasolni.

A Linux alatt lassan szabvánnyá váló karakterkódolás az Unicode UTF-8 (lásd ***Reuven M. Lerner*** Unicode című cikkét a Linuxvilág márciusi számában). Jó hír, hogy ez már a bolygón található valamennyi jelet képes rendszerbe foglalni, így semmilyen más kódolásra nincs szükségünk. A *4. képen* a Red Hat 8.0-hoz kapott Emacs látható, amint az összes szimbólumot és karaktert megjeleníti. Rossz hír viszont, hogy mivel a nem ASCII-karakterek több mint 8 bitet használnak, illetve néha a képernyőn is nagyobb képterület igényelnek, számos mélyen beivódott tézis, például az „1 karakter = 1 byte” egyenlet, egyszerűen megszűnik. Témába vágó forrásmű (a Linux Unicode HOWTO-n kívül) az „UTF-8 soft and hard conversion” mini útmutató és az `UTF8_STRING` szerkezet, aminek célja megőrizni az X vágólap rendszerét az UTF-8 világban. Magasabb szinten az ➲ http://Openi18n.org feladata az operációs rendszertől független, nemzetköziesített programozás.

Menük és ikonok

Mi kellhet még a szép GUI-hoz? Menük és ikonok.

A ➲ http://freedesktop.org-on már most megtalálható a munkafelületek alapszintű szabványa, ami környezettől függetlenül megadja, hogyan kell felépíteni a menüket és elindítani az alkalmazásokat. Igaz, megvannak a maga korlátai, például annak a közös helynek a hiánya, ahová a ***.desktop***-fájlokat be lehetne tenni; illetve a menük beégetése, ami abból a tényből fakad, hogy egyszerűen a merevlemezen található fájlok elhelyezkedését tükrözik. Ezeket a hátrányokat kiküszöbölendő a szabványhoz már készül a virtuáliskönyvtár-bővítmény. Elérhető egy hasonló témájú, másik meghatározás is, ami az ikonhelyeket és a témaválasztást szabványosítja.

A telepítés egyszerűsége

Ha a forrást is terjesztjük, nyilván mindenki lefordíthatja és telepítheti magának a programunkat. De miért nehezítsük meg mások életét azzal, hogy nekik kelljen rájönniük, miért is nem találja az a program a feltelepített programkönyvtárakat? Miért égessük be a dolgokat, ha azok így csak egyetlen terjesztésen fognak működni? Bármilyen alkalmazást is szeretnénk írni, az LSB csoport Linux fájlrendszerszerkezete jól jöhet ilyenkor.

Összegzés

Semmi bajom a tiszta KDE vagy tiszta Gnome rendszerekkel. Mindössze csak azt remélem, hogy a jövő alkalmazásaiban egyszerűbb lesz bármilyen programkombinációból a saját környezetünket összeállítani, a valós szolgáltatások és a teljesítmény feláldozása nélkül is. Az itt leírt módszerek és eszközök sokat segíthetnek egy ilyen alkalmazás megírásakor, fejlesztőiket köszönet illeti.

Köszönetnyilvánítás

Köszönetet szeretnék mondani még ***Havoc Penningtonnak***, ***Keith Packardnak***, a kde-devel lista tagjainak és mindenkinek, aki válaszolt a linuxjournal.com weblapra, és ezzel hozzájárult e cikk megszületéséhez.

Linux Journal 2003. április, 108. szám

A kapcsolódó címek a CD Magazin/Munkafelület könyvtárában találhatóak.

Marco Fioretti
Alkatrész-rendszermérnök, aki sokat foglalkozik a szabad programokkal az EDA-felület és hatékony asztali rendszerek formájában. Marco családjával Rómában él.

A Gnome 2 munkafelület

Vessünk egy pillantást a Gnome 2 munkafelület újdonságaira és képességeire!

A Gnome Alapítvány (gnome.org) a múlt nyáron kiadta a Gnome 2.0-s változatát, amelyet 2003 januárjában a 2.2-es változat követett. Ezek a kiadások a szabványosított munkafelület és a rendszeres kiadások irányába mutatnak. A Gnome immár kitűnő választássá vált a területtel még csak most ismerkedő, nem szakértő felhasználók számára. „Néhány különleges alkalmazás kivételével a Gnome asztal segítségével bárki tökéletesen eredményes lehet. Ez olyasmi, ami csak kevés szabad programmal történt meg az elmúlt 12–18 hónapban” – mondta ***Tim Ney***, a Gnome Alapítvány vezérigazgatója. A Gnome 2 újításainak megjelenésével a Linux esélyei piaci részesedésének növelésére jelentős mértékben megnövekedtek.

Változások a Gnome-hoz képest

Az átállás a Gnome 1.x változatáról a Gnome 2-re néhány apróbb bosszúsággal jár, ami elfogadható, ha munkafelületet fejlesztünk. Az egyik tanulási folyamatról a másikra lépni általános dolog Linux alatt; mégis, amikor a saját gépemet Gnome 2-re állítottam át, meglepetten vettem észre, hogy a Gnome 1.4 olyan kényelmet biztosított számomra, amit nem szerettem volna megzavarni.

A Gnome Alapítvány munkatársai elmondták, hogy korszerűsíteni szerették volna az asztalt, csökkentve egy kicsit a választási lehetőségeket, így a munkaállomások egyszerűbbé válnak és csökken az új felhasználók tanulási ideje. Ennek eredményeképpen számos alkalmazás, különösen az ismétlődő darabok vagy teljesen eltűntek, vagy bekerültek az ***Egyebek*** (Extras) almenü alá. A Gnome 2 egységesebb kinézetet (look and feel) nyújt a programok között, hála a témákban és a karakterkészletek terén végrehajtott fejlesztéseknek. „Próbálunk jobban odafigyelni az egyensúlyra – szabványokat állítunk fel, miközben megőrizzük a Linux rugalmasságát” – mondta ***Havoc Pennington***, Gnome-fejlesztő és alapítványi tag.

Meglehet, hogy sokan nagyon elégedettek vagyunk a Linuxszal úgy, ahogy van, de az avatatlan felhasználók nem feltétlenül értékelik a küszködést. A nagyobb piaci térnyerés érdekében a Gnome leegyszerűsítette az asztalát. Pennington így fogalmazott: „Nem arról van szó, hogy eltávolítjuk a lehetőségeket, pusztán olyan alapértelmezéseket adunk meg, amelyek segítségével gyorsan és könnyen elkezdhetjük a munkát.” Így ahelyett, hogy öt különböző böngészőt és három szövegszerkesztőt tennénk elérhetővé, mindegyikből csak egy lesz. Ha nem kedveljük a kiválasztottat – amelyet az új Gnome 2 könyvtárral terveztünk –, még mindig letölthetjük és feltelepíthetjük a Webről a megfelelő RPM-et.

A Gnome Projekt ***Seth Nickell*** által vezetett felhasználhatósági tanulmányának hála, a programmenüket szépen újrarendszereztük. Mostantól a Gnome nem ás el eszközöket az almenürétegek alá vagy más programokba. Arra sincs szükség, hogy a felhasználó parancssori változtatásokat végezzen vagy hogy közvetlenül megváltoztassa a beállítás-parancsfájlokat. Ehelyett a Gnome 2 szinte az összes rendszerbeállítást grafikus felülettel végzi el, amik a főmenük alatt könnyen megtalálhatók.

A Gnome számos alkalmazást és eszközt tartalmaz. Mivel

1. kép Gnome asztal

képtelenség valamennyit bemutatni, minden csoportból csak néhány kulcsfontosságú elemet emelek ki, megpróbálva segíteni azoknak, akik most ismerkednek a Gnome-mal. Néhány elem újdonság a Gnome 2-ben, míg mások már egy ideje megtalálhatók a Gnome rendszereken.

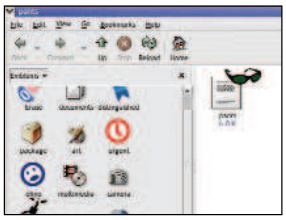
Panelek

Mint az már néhány munkakörnyezetben elterjedt, a Gnome is panelek segítségével teszi lehetővé felhasználóinak, hogy programokat indíthassanak vagy kezeljenek, illetve megfigyelhessék a rendszerüket. A paneleket az asztal felső, alsó, bal vagy jobb margójára helyezhetjük. Egy margóra több panelt is felrakhatunk, illetve a panelek lebegők is lehetnek, így akárhová helyezhetők az asztalon. Beállíthatjuk, hogy maradjanak nyitva vagy önműködően rejtőzzenek el, illetve az éppen futó alkalmazások számától függően átméretezhetik magukat. A panelre gombot is szerelhetünk, amivel kiereszthetjük vagy visszahúzhatjuk azt. Kezdetben a felső margón találunk egy panelt, ami a programok indításához szükséges menüket tartalmazza. Alulra egy másik panel kerül, amin az OpenOffice.org fontosabb alkalmazásainak indítására szolgáló ikonok kaptak helyet, ugyanis ez lett a Gnome 2 hivatalos irodai csomagja. Hivatkozást találunk itt a Mozilla böngészőre, valamint az alapértelmezett levelezőre, a Ximian Evolutionre. Mindent egybevetve ezek az eszközök jó választást jelentenek egy átlagos Linux-felhasználó számára, különösen, ha járatlan felhasználóról van szó. Ha a kiválasztott elemek nem tetszenek, a panelekről könnyedén leszedhetünk, illetve felrakhatunk új programindító ikonokat. Eltávolításhoz egyszerűen kattintsunk a jobb gombbal az ikonra, és válasszuk a ***Törlés a panelről*** (Remove from the panel) utasítást. Az elemek hozzáadásához a jobb gombbal mutassunk a panel egyik üres pontjára, válasszuk a ***Panelra helyezés*** (Add to panel) menüpontot, majd az almenükből keressük ki a beilleszteni kívánt programot.

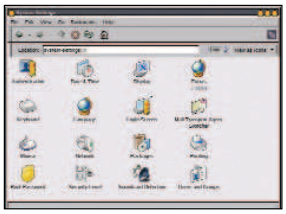
Munkaterületek

Indításkor az alsó panelen egy munkaterület-váltót, más néven lapozót találunk. Bár nem Gnome-találmány, a munkafelület

2. kép A Gnome 2.2 betűtípusai tényleg szebbek



3. kép Képernyőkép



4. kép Nautilus-rendszerbeállítások

hasznos eleme: lehetővé teszi, hogy egyidőben több virtuális munkafelületet futtassunk. Az egyik lapon megnyithatunk néhány egymáshoz tartozó alkalmazást, miközben a másik lapon más, szintén egymáshoz tartozó alkalmazásaink futnak, így gyorsan válthatunk közöttük, anélkül, hogy az ablakokat maximalizálni vagy minimalizálni kellene. Ha például olyan programot használunk, ami minden képet és eszközt külön ablakban nyit meg (ilyen például a The GIMP), jól jön, ha a Gimp ablakainak megzavarása nélkül van lehetőségünk a másik munkaterületre váltani, ahol megnézhetjük a leveleinket.

Kisalkalmazások

Számítógép-rendszerünktől függően elképzelhető, hogy bizonyos kisalkalmazások már eleve fent vannak a paneleken, de magunk is felhelyezhetünk újakat. Alapértelmezés szerint kezdetben csak az óra-kisalkalmazás él. Ugyanakkor a kinézetválaszték most már kizárólag egyetlen digitális változatra szűkült. A régi Gnome majd fél tucat változatot nyújtott, többek közt néhány stílusos analóg órafelületet is. Laptopok esetében a panelre az elemfigyelő kisalkalmazás is felkerül. Felrakhatunk még szótárűrlapot, időjárás-előrejelzőt, görgő részvény-árfolyammutatót, modem-jelzőfényeket, levelesláda-jelzőt, CD-lejátszót és hangszabályzókat, valamint hajlékonylemez-befűzőt (ami hasznos lehet azoknak a felhasználóknak, akik nem ismerik a `mount` parancsot, és adatokat szeretnének menteni).

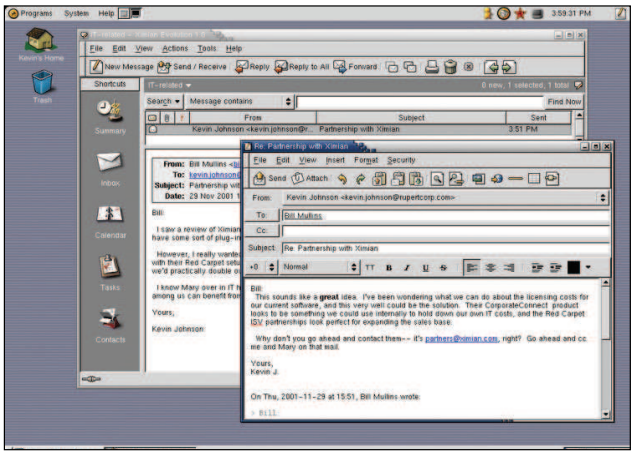
Egységesség

Az egységesített asztali környezetek egyik hasznos képessége, hogy az alkalmazások között szöveget tudunk másolni és beilleszteni. Az utóbbi években ezt a kényelmet már a Gnome alatt is élvezhettük, csak sajnos elég gyengécske és nem túl megbízható változatban. Szerencsére úgy tűnik, ez többé már nem lesz gond. Mostantól másolhatunk böngészőből a `vi`-t futtató terminálablakba vagy szövegszerkesztőbe és így tovább. E hiba kiküszöbölése nagymértékben hozzájárul a Gnome-környezet egységességének erősödéséhez.

A másik egységesítési fejlesztés a képernyő-karakterkészletek fejlesztése. Korábban a karakterek elég fűrészfogasan néztek ki. Mindez megváltozott, vagy ahogy az ausztrál ***Jeff Waugh***, sydney-i Gnome Release koordinátor fogalmaz: „A GTK+ 2.0 (amit a Gnome 2.0 használ) alkalmazásával hozzájutottunk a Pango betűtípustervezet-megjelenítő és i18n könyvtárhoz. A GTK+ 2.2 már egy új, `fontconfig` nevű betűtípusbeállító programot használ, amelyet ***Keith Packard*** írt. Ezáltal tényleg letisztulnak a betűtípusok. A változás a Gnome összes asztali alkalmazására kiterjed, hiszen valamennyi GTK+ alapú. Az eredményül kapott munkafelület egységes, profi kinézetű lesz.”

Nautilus

A Gnome beépített része a Nautilus nevű grafikus felület, ami a fájlkezelést és a Linux beállítását hivatott elvégezni. A Nautilus elérésének a legegyszerűbb módja az, ha kétszer kattintunk az asztal ***Home*** ikonján. A Nautilus egységes húzd és ejtsd alapú fájlkezelő. A fájlokat másolhatjuk és mozgathatjuk billentyűleütésekkel, vagy úgy, hogy a fájlikont az egyik ablakból áthúzva a másikba ejtjük, illetve ha a jobb gombbal kattintunk a fájlikonra és a helyi menüt vesszük igénybe. A helyi menü egyúttal a fájl tulajdonosának és jogosultságaink a módosítására is felhasználható. Bizonyos fájlokhoz grafikus jelzéseket rendelhetünk, felcímkézve őket (például fontos vagy személyes jelzésekkel).



5. kép Ximian Evolution

Azok, akik hozzám hasonlóan adatfájlok százait gyűjtik össze a számítógépükön és könyvtárak tucatjait ássák jó néhány szinten keresztül, biztosan értékelni fogják a Nautilus könyvjelző képességét. Egyszerűen csak felkeressük a gyakran használt könyvtárat, rákattintunk a ***Könyvjelzők*** (Bookmarks) legördülő menüre és kiválasztjuk az ***Könyvjelző hozzáadása*** (Add Bookmark) parancsot. Ha legközelebb ebbe a könyvtárba szeretnénk belépni, mindössze csak rá kell kattintanunk a ***Könyvjelzők*** menüben létrejött ikonra, és máris ott termünk.

Beállítások

A Nautilus egyúttal grafikus csatolófelületként használható a Gnome és az alatta található operációs rendszer beállításához. A Gnome-eszközöket és -beállításprogramokat megtaláljuk a menükben, vagy a Nautilus ***Ugrás*** (Go) elnevezésű menüjéből a ***Kezd itt!*** (Start Here) pontra kattintva. Az ablak négy programcsoportikonnal jön fel, ezek neve: ***Alkalmazások*** (Applications), ***Munkaasztal-beállítások*** (Desktop Preferences), ***Kiszolgáló-beállítások*** (Server Settings) és ***Rendszerbeállítások*** (System Settings). Mellesleg az 1.4-es alatt az asztalon külön ikon volt erre a szolgáltatásra, ezt azonban most a ***Munkaasztal beállításai*** menük alatt találjuk meg.

- Az ***Alkalmazások*** menücsoport a menüpanel főmenüjében található valamennyi alkalmazást összekapcsolja. Itt indíthatjuk el a programokat vagy új alkalmazásindítókat adhatunk a menükhöz. Ez a szolgáltatás azonban nem működik a Red Hat 8.0-val szállított 2.0-s változat alatt. A 2.2-es változat alatt azonban a hibát kijavították.
- A ***Munkaasztal beállításai*** menücsoport alatt számos különféle beállítást módosíthatunk, többek között a hátteret, az alapértelmezett betűtípust és az egér beállításait. Másik témát választhatunk vagy az ablakok fókuszálási módját is

átállíthatjuk. Akik inkább billentyű-, semmint egérkedvelők, értékelni fogják a ***Gyorsbillenytűk*** (Keyboard Shortcuts) eszközt. Itt olyan billentyűkombinációkat hozhatunk létre, amelyekkel megnyithatjuk a kedvenc programjainkat vagy munkaterületet válthatunk. Sok minden eleve be van állítva, de bármit megváltoztathatunk.

- A ***Kiszolgálóbeállítások*** programcsoport az Apache és a hasonló kiszolgálóalkalmazások beállításához tartalmaz hivatkozásokat. Ez a programcsoport több-kevesebb eszközt tartalmazhat, attól függően, hogy mit telepítettünk a rendszerre. Legszűkebb kiépítésben csak a */etc/rc.d/init.d* könyvtárban található `xinetd` szolgáltatáshoz találunk itt beállítófelületet. Ezek ugyanazok a rendszerszolgáltatások, amelyeket a régi beállítóprogramból elérhettünk.
- A ***Rendszerbeállítások*** programcsoport tartalmazza a számítógép beállításához tartozó összes hasznos dolgot. Közülük számos eszköz a Red Hat része, de a Gnome-hoz fejlesztették. Ilyen például a dátum- és időeszköz felülete és a videómegjelenítés megváltoztatását végző eszköz. Egyébként mostantól itt tudjuk megváltoztatni X-beállításainkat is; többé már nem a beállítóprogram (setup) része – ha esetleg csalódottan tapasztaltuk volna, hogy ott nem találjuk az X-beállítót. A 2.2-es változattal a több videokártyával és monitorral megvalósított többfejes megjelenítés támogatása is leegyszerűsödött. Waugh szerint „a Nautilus egyetlen folyamata kezeli mindkét munkafelületet, és a panelek mindkét fejen megjelenhetnek stb. Még olyan szolgáltatások is akadnak, amivel az alkalmazásokat a megjelenítők között áttehetjük".

Ha a ***System Settings*** ablakban a ***Network*** ikonra kattintunk, egy (igen tetszetős) eszköz nyílik meg, amivel beállíthatjuk a hálózati kártyáinkat és a ***hosts*** fájlt. A nyomtatóeszköz (printconf-gui) lehetővé teszi, hogy a felhasználó nyomtatómeghajtókat állítson be, illetve újraindítsa a nyomtatódémont.

Kezelhetőség

A Gnome 2-ben végrehajtott fejlesztések közül a szabad programok piaci részesedésének növekedésére a legnagyobb hatással a csökkent képességű embereket segítő kezelőszolgáltatások megjelenése lehet. „Az Egyesült Államok kormánya mostantól nyílt forrású munkafelület-megoldásokat alkalmaz, ami egyébként nem történhetett volna meg – a kormányzati szabályozások miatt" – hangoztatta Pennington. Majd hozzátette, hogy a többi felhasználónak is sokféleképpen hasznára lehetnek ezek a szolgáltatások. Például az egyik kezelhetőségi követelmény a billentyűzettel történő teljes irányíthatóság (mint már fentebb említettünk). Immár szinte bármit meg tudunk csinálni a billentyűzetről is. A kezelhetőségi szabályozásoknak megfelelően a témák is fejlődtek a színkontraszt, az alapértelmezett betűtípusméret stb. terén."
A kezelhetőségi korlát megszűnésével a Linux célpiaca nagymértékben kiszélesedett, ugyanakkor egyúttal valamennyi felhasználó lehetősége is bővült. Pennington szerint „a Sun Microsystems vitte elsőként a kezelhetőséget a Gnome-megoldásokba. Hatalmas projekt volt, ami pár évet és húsz fejlesztőt vett igénybe. Nagyon örülünk ennek az eredménynek, és büszkék vagyunk arra, amit elértünk."

Menük és alkalmazások

A fent említett rendszerbeállító menükön kívül a főmenüben néhány más menüpontot is találunk, ezek alkalmazások elindítására szolgálnak. A ***Kellékek*** (Accessories) nevű menüpont számológépet, szótárat és egy egyszerű szövegszerkesztőt (gedit) tartalmaz. Az Office címke alatt találjuk az OpenOffice.org-ot és a Diát, amellyel szervezeti diagramokat és folyamatábrákat készíthetünk a projektek kezelésére tervezett MrProject mellett. A ***Grafika*** (Graphics) menüpont alatt érhetjük el a Gimpet és néhány más képalakító programot. A ***Játékok*** (Games) menü sok játékot tartalmaz, néhányuk közülük használja is a Gnome-könyvtárakat: pár kártyajáték, néhány népszerű „egy sorba a pontokat" játék, az aknakereső Gnome-változata, a Mahjongg és további kedvencek. Nem sorolhatom fel az összes alapértelmezés szerint feltelepített alkalmazást, de gondolom, látható, hogy a Gnome-menük korszerűsítése nem téveszti meg a felhasználókat.

Ximian Evolution

A Gnome-könyvtárakat kihasználó alkalmazások közül talán a legkiválóbb darab a Ximian levélügyfele, az Evolution. 2002 novemberében a Linux Journal az olvasók Linux alatt elérhető egyik legkedveltebb levélügyfeleként említette meg. Az Evolution több POP- és IMAP-azonosítót támogat egyszerre. Tartalmaz levélszűrést, helyesírás-elemzőt, és bináris állományok csatolására is lehetőséget nyújt. Bár alapesetben a hagyományos POP- és IMAP-kiszolgálókkal működik, a Ximian üzleti Connectorával a felhasználók kapcsolódhatnak a Microsoft Exchange kiszolgálókhoz és fogadhatják a csoport-címjegyzékeket, valamint használhatják a megbeszéléstervezést – ami fontos együttműködési lehetőség.
Az Evolution egyik különleges képessége, amit nem találunk meg más levelezőprogramban, a virtuális könyvtárak használata. „A V-Mappák könyvtárak nagyobb rugalmasságot engednek a felhasználóknak és megkönnyítik a levelek rendezését. Ez valami olyasmi, ami kizárólag az Evolution sajátja"– mondta ***Christine McLellan***, az Evolution vezető termelési menedzsere. Például azon felül, hogy megnézhetjük a bejövő levélládánkba érkezett leveleket, amelyeket dátum, cím vagy feladó szerint rendezhetünk, a virtuális mappák képesek arra is, hogy az összes olvasatlan levelet megmutassák. Ezzel a virtuális könyvtárral gyorsan és egyszerűen végigfuthatunk az összes üzeneten. Virtuális könyvtárakat készíthetünk az adott címmel vagy a bizonyos emberektől érkező levelekhez is. Figyelnünk kell azonban arra, hogy ha egy üzenetet valamelyik virtuális könyvtárban kitörlünk, egyúttal az összes többi mappából is törlődik.

Összegzés

A Gnome Alapítvány legendás fejlesztéseket végzett az asztalon és a Gnome 2 megérkeztével végre elérték az olyannyira hiányolt egységességet és üzembiztonságot. „Mindez nem lett volna lehetséges fejlesztőink nélkül, akiknek százait nem a Gnome fizette, és alkalmazóik sem támogatták őket"– mondta Ney. Jelenleg az Alapítvány tervei szerint minden hatodik hónapban új változat jelenik majd meg, amiből 2003 júniusára várható a következő (v. 2.4). A 2.4 tartalmazni fogja a Nautilus fogd és ejtsd CD-író szolgáltatását és a további betűtípus-fejlesztéseket. Elkötelezvén magát az asztal rendszeres frissítése mellett, a Gnome olyan munkafelületté vált, amire üzletemberek, felhasználók és fejlesztők bátran alapozhatnak.

Linux Journal 2003. április, 108. szám

Russell Dyer (russel@dyerhouse.com)
Perl-programozó és MySQL-fejlesztő. Tanácsadói munkakörben dolgozik New Orleans körzetében. Angolból szerzett diplomát és szakdiplomáját (Master's degree) szintén angolból készíti.

Linux a kisvállalkozásokban

Ha egyéni vállalkozó vagy, esetleg egy kis üzletet vezetsz, mutatunk néhány irodai alkalmazást, amit használhatsz.

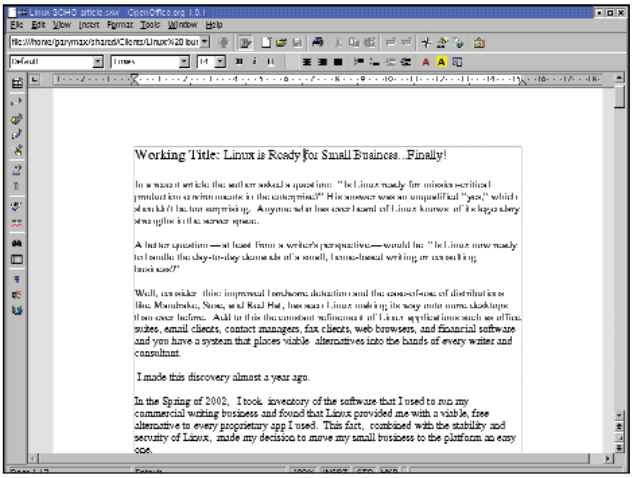
Nemrég egy cikkben a szerző feltette a kérdést: „Készen áll-e a Linux a kulcsfontosságú vállalati termelőfolyamatokban való felhasználásra?" A válasza határozott igen volt, ami nem túl meglepő. Bárki, aki valaha hallott a Linuxról, tudja, hogy a kiszolgálóként legendásan jól teljesít. Jobb kérdés lenne – legalábbis az én szemszögemből nézve – a következő: „Készen áll-e a Linux az IT-személyzetet nem foglalkoztató kisvállalkozások szokásos feladatainak az ellátására?" A továbbfejlesztett alkatrész-felismerés és a könnyen használható terjesztések (például Mandrake, SuSE és Red Hat) lehetővé tették, hogy egyre több munkaasztalon jelenjen meg a Linux. Ehhez még hozzávehetjük, hogy a Linux-alkalmazások, például az irodai csomagok, a levelezőprogramok, a névjegykezelő programok, a faxprogramok, a webböngészők és a könyvelőprogramok állandóan fejlődnek, és ezek együtt olyan rendszert alkotnak, ami az átlagos irodai vagy tanácsadó munkát végző kisvállalkozások számára valódi választási lehetőséget jelent.
Ezt a felfedezést csaknem egy éve tettem. 2002 tavaszán leltárt készítettem az írói munkámhoz felhasznált programokról, és rájöttem, hogy a Linux valódi, ingyenes választást kínál minden általam használt zárt programra. Ez a tény, valamint a Linux megbízhatósága és biztonságossága arra döntésre sarkallt, hogy térjek át a használatára. Azóta sem bánom ezt a döntést.
Talán te is író vagy tanácsadó vagy, és komolyan gondolkozol a lehetőségeken, akárcsak én tettem. Az is lehet, hogy hallottál a Linuxról, de nem vagy biztos benne, hogy milyen alkalmazásokat használj. Írásomban négy, írók és tanácsadók számára hasznos programot mutatok be, amelyek bármelyik Linux-terjesztésben megtalálhatók. Helyhiány miatt nem adhatok teljes leírást a programokról, de reményeim szerint eleget mondok el ahhoz, hogy olvasóim megismerjék az alkalmazások felhasználási területeit a kisvállalkozásokban.
A tárgyalt Linux-alkalmazások nem feltétlenül rendelkeznek zárt párjuk összes tudásával. A Linux-környezetben fellelhető alkalmazások az alapfeladatok ellátására ugyanúgy alkalmasak, de nem szükségszerűen rendelkeznek ugyanazokkal a csingilingikkel. Ne törődjünk most a csingilingikkel, inkább arra összpontosítsunk, hogy ezekkel az eszközökkel aránylag olcsó és megbízható, biztonságos környezetben végezhetjük el feladatainkat.

Az eszközök

A szakírói vagy tanácsadói kisvállalkozás nem támaszt nagy igényeket a programokkal szemben. Azt hiszem, ez az egyik nagy előnye az ilyen típusú vállalkozási formának. Ezek a cégek jellemzően a következő alkalmazásokat használják: irodai csomag, névjegykezelő, illetve levelezőprogram, webböngésző és könyvelőprogram. Minden más csak hab a tortán.

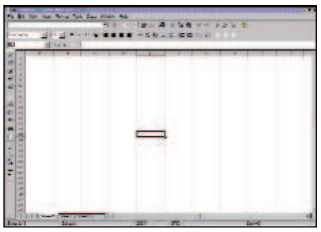
OpenOffice.org

Irodai csomagnak az OpenOffice.org programot javaslom, mert nyílt forrású, több operációs rendszer alatt is fut, és szabadon hozzáférhető a ➲ http://www.openoffice.org címen. Az OpenOffice.org csomag egyik nagy előnye az XML alapú állományformátum használata. Az XML olyan szerkezetben tárolja az adatokat, hogy azok különböző számítógéprendszerek között is cserélhetők maradnak, így adataink hosszú évek múltán is hozzáférhetőek lesznek.
Az OpenOffice.org csomag négy alkalmazásból áll: OpenOffice.org Writer, OpenOffice.org Calc, OpenOffice.org Impress és OpenOffice.org Draw.
Az OpenOffice.org Writer a csomag szövegszerkesztője. A felhasználói felület ismerős lesz, hasonló a Microsoft Word vagy a Sun StarOffice Writer felületéhez (*1. kép*).

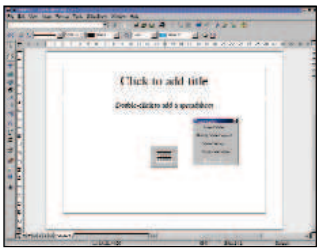


1. kép Az OpenOffice.org Writer

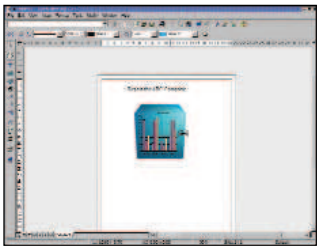
Egy író általában szövegszerkesztője tudásának csupán 10–15 százalékát használja ki, ugyanis a formázás és a grafikai munkálatok nagyobbik részét a megrendelő végzi. Akár üzleti levelekről vagy ajánlatokról, akár cikkekről vagy könyvekről van szó, az OpenOffice.org Writer több mint elég a munkám elvégzéséhez.
A Writer felhasználói felülete magától értetődő, és könnyen megtalálom benne, amit keresek. Mindent tud, amire manapság egy szabadúszó írónak, illetve tanácsadónak szüksége lehet, sőt még többet is. Számos állományformátumot ismer, beleértve a következőket: Microsoft Word 6, Microsoft Word 95, Microsoft Word 97/2000/XP, RTF (Rich Text Format) és StarWriter 5 (a régebbi StarOffice-állományformátum). Képes PDF-be exportálni.
Vigyázat, ha sok táblázatot vagy különleges formázásokat használunk, ezek bizonyos mértékben torzulhatnak vagy elveszhetnek, már ha Microsoft Wordbe vagy egy másik irodai csomag állományformátumába exportáljuk a dokumentumot.
A szabály az, hogy minél egyszerűbb a dokumentum formázása, annál jobb az átvihetősége egy másik szövegszerkesztő alá.



2. kép
OpenOffice.org Calc számolótábla



3. kép Bemutató készítése



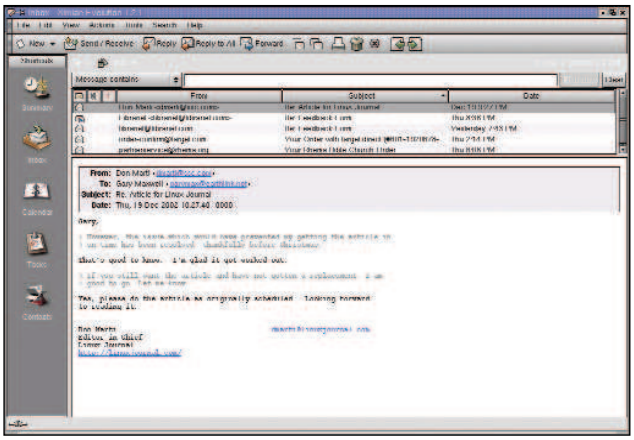
4. kép Grafikon rajzolása

Mivel az íróknak nincsenek nagy igényeik, nem is nagyon használom az OpenOffice.org csomag többi programját. Ami a táblázatkezelőt illeti, az OpenOffice.org Calc tudja mindazt, amit az ember a táblázatkezelőktől elvár: önműködő összegzés, önműködő formázás, grafikonkészítés és sok egyéb. A munkákat menthetjük a Microsoft Excel, a StarCalc, a Data Interchange Format formátumokba és természetesen a Calc saját formátumába is (*2. kép*).
Az OpenOffice.org Impress bemutatókészítő program rendelkezik a Microsoft PowerPoint alapvető képességeivel, de nem tud sablonokat használni. Sablonok nélkül tovább tart a bemutatók megtervezése. Az Impress olvassa és írja a PowerPoint, a StarImpress és a saját állományformátumát (*3. kép*).
A Draw nevű rajzprogram tartalmazza a szokásos rajzeszközöket, és képes a legelterjedtebb grafikusállomány-formátumok beolvasására. Menteni az OpenOffice.org Draw és a StarDraw formátumába képes. Az egyszerű igényeket kielégíti (*4. kép*).

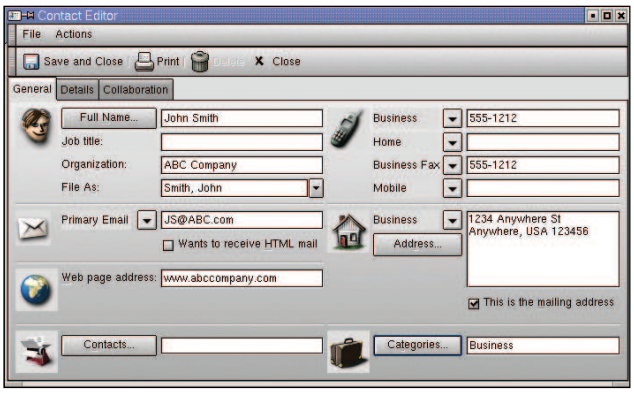
Névjegykezelő és levelezőprogram

Bár a Linux sokféle megoldást kínál a névjegyek tárolására és a levelezés intézésére, én azokat a megoldásokat szeretem, amik mindkettőt egyben tartalmazzák. Ez kényelmesebb, minden rögtön egy helyen van, nem kell két különböző alkalmazást nyitva tartani. A Ximian Evolution kielégíti ezt az igényt. A program egyszerre levelező, névjegykezelő és feladatütemező, amiben az író, illetve tanácsadó egy helyen tárolhatja adatait. A program első indításakor az ***Összefoglaló*** képernyő jelenik meg (*5. kép*). Itt az időjárás és a Linuxszal kapcsolatos legfrissebb cikkek mellett láthatjuk, hogy hány beérkezett és feladásra váró üzenetünk van. Megjelennek be nem fejezett feladataink és az adott napra szervezett találkozóink is. Az ***Összefoglaló*** ablaktól balra elhelyezkedő listában állíthatjuk be a találkozókat és a feladatokat, illetve intézhetjük a levelezést. A ***Beérkezett levelek*** ikonra kattintva léphetünk át az Evolution levelezőrészébe (*6. kép*).
Az Evolution hasonlít a Microsoft Outlookra. A szokásos levelezőműveletek gombjai az ablak felső részén helyezkednek el.

5. kép
A Ximian Evolution elindítása

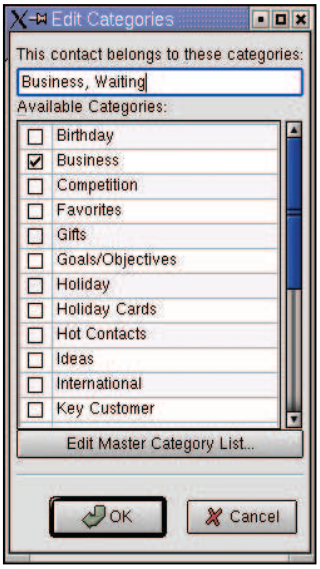


6. kép A beérkezett levelek ellenőrzése

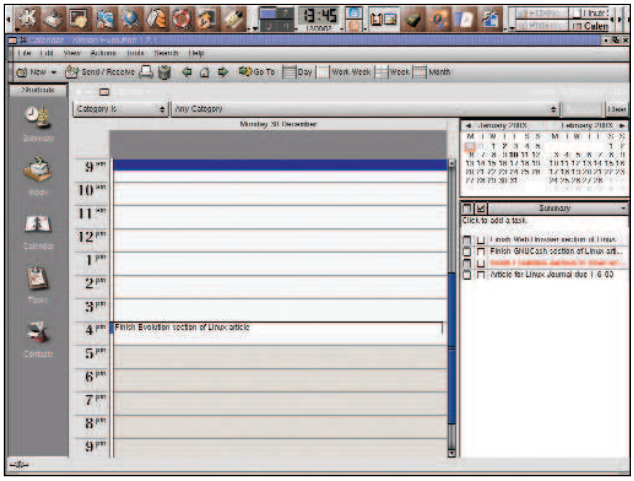


7. kép Névjegy az Evolutionben

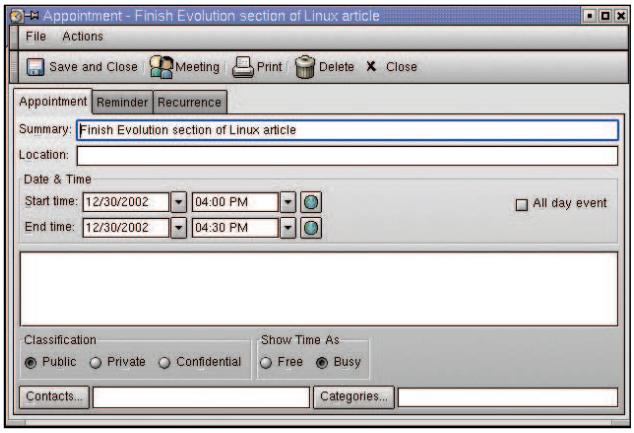
Az Evolution 1.2.1 egyik hasznos újítása az ***Új*** gomb. Segítségével egy kattintással lehet új levelet vagy új névjegyet létrehozni, anélkül, hogy az Evolution megfelelő részébe be kellene lépnünk.
Könnyű névjegyeket hozzáadni és kezelni az Evolution névjegykezelőjével (*7. kép*). Beírhatjuk a szokásos adatokat (név, cím, telefonszám stb.), de a ***Részletek*** fülre kattintva még több adat felvételére nyílik lehetőség. Minél több adatot gyűjtünk össze az ügyfelekről, annál könnyebb együttműködni velük. Az ***Együttműködés*** lapon megadhatjuk azt a címet, amelyen az ügyfél a határidőnaplóját közzétette az Interneten. Ez újabb példája annak, hogy hány jó ötletet valósítottak meg az Evolution fejlesztői.
A névjegyek felvétele után a listában tetszőleges számú feltétel szerint kereshetünk: betűrend, elektronikus levélcím vagy kategória (*8. kép*) alapján is kutakodhatunk. Saját keresési feltételeket is létrehozhatunk a ***Haladó*** lehetőség választásával.
A névjegyeket tetszés szerinti



8. kép Névjegy besorolása



9. kép Az Evolution Naptár



10. kép Találkozó

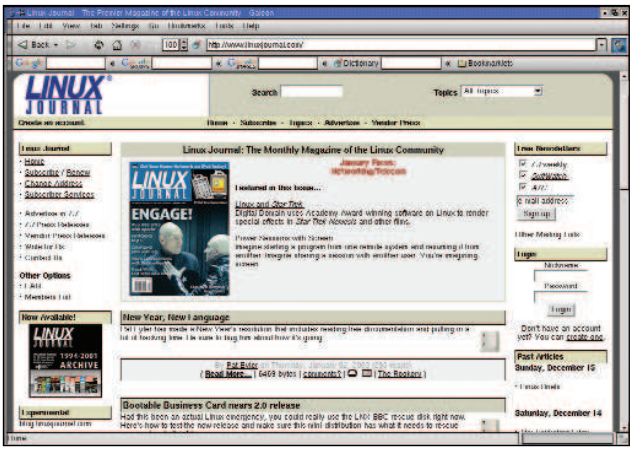
11. kép A Mozilla

kategóriákba sorolhatjuk be, és eszerint kereshetünk rájuk. A ***Naptár*** (*9. kép*) használata eléggé egyértelmű, a naptárablakot pedig ízlés szerint be lehet állítani. Az ablaktáblák a határvonalak huzigálásával átméretezhetők. A kívánt nap kiválasztása után a találkozó idejére duplán kattintva egy ablak nyílik meg (*10. kép*). Itt adhatjuk meg a találkozó részleteit. Figyelmeztetést is beállíthatunk a találkozó előttre néhány perccel, órával vagy nappal.

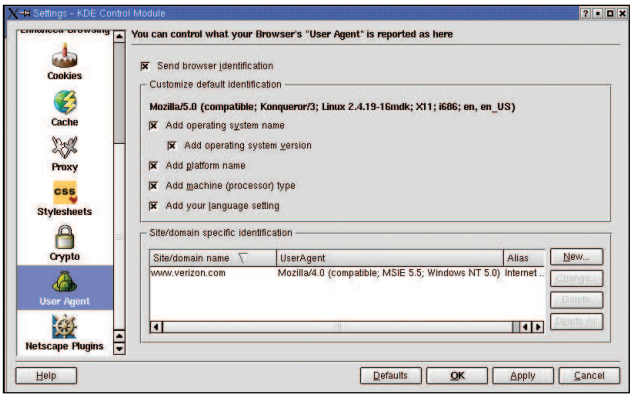
A ***Feladatütemező*** hasonlóan működik. Az ***Új*** gombra kattintva válasszuk a ***Feladat*** elemet. Megnyílik egy ablak, amiben megadhatjuk a feladat részleteit. A feladatok az Evolution ***Feladatok*** részében jelennek meg – időrendben és feladatcsoportok szerint is kiírathatók. A befejezett feladatokat bejelölhetjük, így könnyedén követhető, mivel végeztünk és mi maradt még hátra. Még számos más önálló felfedeznivaló van az Evolutionben. Akárcsak a Linux, az Evolution is lehetővé teszi, hogy egy feladatot többféleképpen végezzünk el.

Webböngésző

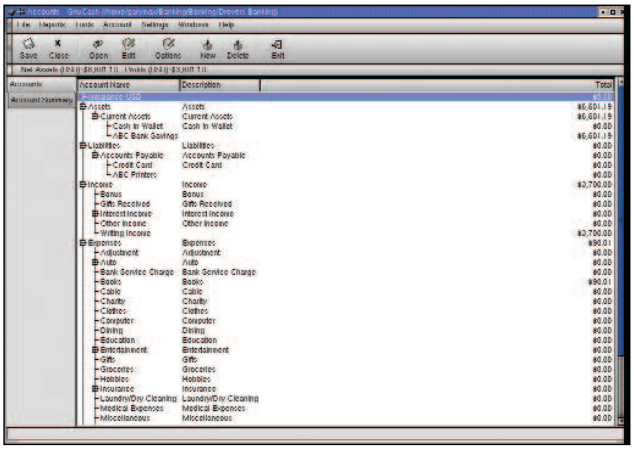
A két legnépszerűbb böngészőprogram a Mozilla (*11. kép*) – ami ugyanarra a megjelenítőmotorra épül, mint kereskedelmi párja, a Netscape – és a Galeon (*12. kép*), ami sok tekintetben hasonlít a Mozillára, és része a Gnome munkafelületnek.
E kettő mellett a Linuxban megtalálható még a Konqueror, a KDE alapú fájlkezelő és böngésző. A legtöbb linuxos böngésző támogatja a biztonságos kapcsolatok 128 bites titkosítását. Általában érdemes egynél több böngészőt telepíteni. Az ok a webhelyek elérése. Először az gondolhatnánk, hogy a böngésző az böngésző; a kinézeten és használaton kívül lényegében egyformák. Bármilyen népszerű, a Mozillából hiányzik egy dolog, ami a Konquerorban megvan: az azonosító megváltoztatásának lehetősége. Amikor csatlakozunk egy webhelyhez, a böngésző azonosítja magát, elküldi a nevét, a változatszámát és az operációs rendszer típusát. Bár a legtöbb webhely nem foglalkozik ezekkel az adatokkal, van néhány, ami gondot okozhat, ha nem egy adott típusú böngészővel nézzük meg. Például akárhányszor próbáltam belépni a telefontársaságom



12. kép A Galeon



13. kép A Konqueror azonosítójának beállítása



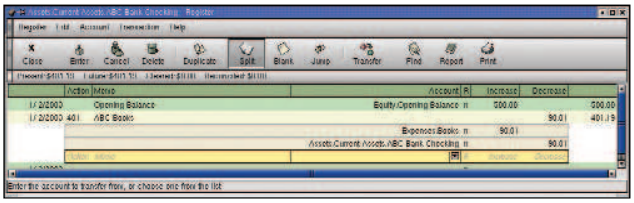
14. kép A GnuCash

15. kép A GnuCash beállítása

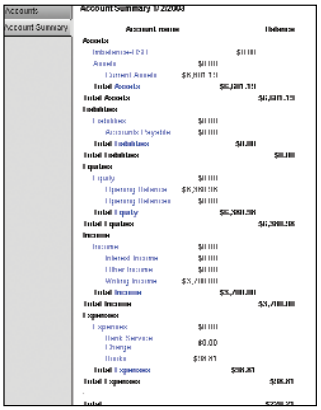
honlapjára, mindig gondjaim akadtak. A honlap egy bizonyos ponton túl nem engedett tovább. Kis gondolkodás után visszaemlékeztem, hogy bizonyos webhelyek nem támogatnak mást, csak a Microsoft Internet Explorert. Itt válik érdekessé az azonosító (*13. kép*). Egyszerűen megváltoztattam a böngésző személyazonosságát Internet Explorer 5.0-ra, és a webhely csont nélkül működött. Nem sokszor jön elő ez a nehézség, de ha mégis, próbálkozz az azonosító átírásával. Ezek a példák szintén azt bizonyítják, hogy a Linux egy feladatra több megoldást is kínál, és mindig akadnak eszközök a zárt megoldások által támasztott akadályok leküzdésére.

Könyvelőprogram

A pénzt megkeresése után szeretnénk nyilvántartani. A könyvelőprogramok ezt az igényt elégítik ki. A Linux egyik legjobban őrzött titka a nyílt forrású könyvelőprogram, a GnuCash (*14. kép*). A GnuCash megbízható, könnyen használható könyvelőprogram, amivel villámgyorsan elvégezhetjük a könyvelési munkát. A GnuCash beállítóvarázslójával egy pár kérdésre válaszolva több számlát is létrehozhatunk, különböző nyitóegyenlegekkel (*15. kép*).
A GnuCash rendelkezik kereskedelmi megfelelőinek a tudásával, például képes a műveletek önműködő befejezésére és a csekkszám növelésére. A GnuCash kettős könyvelést használ, csakúgy, mint a profi könyvelőprogramok.
A kettős könyvelés alapötlete, hogy minden művelet két számlára van hatással, a „tartozik” és a „követel” számlára. Így mindig tudjuk, honnan jött a pénz és hová ment el. Ha például egy, a vállalkozásommal kapcsolatos könyvet szeretnék venni, az



16. kép Hová tűnt a pénz?



17. kép A számlák testreszabása

árával megterhelem a megtakarítás számlámat, és jóváírom a könyvek vételére fenntartott kiadásszámlámat (*16. kép*). Ez a segédeszköz nélkülözhetetlen, ha meg szeretnénk tudni, hová tűnik a pénzünk. A GnuCash nagy előnye, hogy könnyen az egyéni pénzügyi helyzethez igazítható. Léteznek beépített számlatípusok – kiadás, bevétel, értékpapírok, befektetések –, mindegyik saját alszámlarendszerrel bír. Emellett saját számlák is hozzáadhatók, a nem használt számlák pedig törölhetők (*17. kép*). A GnuCash másik jó tulajdonsága, hogy a műveletek szétvághatók, így egy művelet több „tartozik” és „követel” számlát is érinthet. Így a műveletet finomhangolhatjuk, és egy adott pénzügyi műveletben érintett összes számla megjeleníthető.
Végül a GnuCash grafikus jelentései segítségével egy szempillantás alatt áttekinthetjük a nettó értéket, a bevételt, a kiadást, a profitveszteséget stb. Ezek elősegítik pénzügyeink nagy pontosságú kezelését. Sok más lehetőség is van még, de helyhiány miatt nem tárgyalhatjuk őket. Elég az hozzá, hogy a GnuCash mindent tud, ami egy egyszemélyes vállalkozásban kellhet, még többet is.
A cikk írásának idején a GnuCash az 1.6.7-es változatnál tart, és a személyes pénzügyek kezelésére alkalmas. Nincs lehetőség a vevők és a beszállítók nyomon követésére, számlázásra és a számlák kifizetésére. Egy kisvállalkozás könyvelése számára azonban egy kis munkával használhatóvá tehető, mint ezt már fent is említettük. A hamarosan megjelenő GnuCash 1.8-ban a kisvállalkozások könyvelésére is lehetőség nyílik majd, a számlázást és a számlafizetést is beleértve.

Összegzés

Ha jelenleg zárt operációs rendszert és zárt alkalmazásokat használsz, és úgy döntöttél, hogy biztonságos, megbízható és szabad Linux alapú megoldásokra váltasz, akkor gratulálok, és a küldetésem sikeres volt.

Linux Journal 2003. április, 108. szám

Gary Maxwell (gary@garymaxwell.com)
15 évet töltött el egy Fortune 100 vállalatnál, ezután a saját lábára állt és szabadúszó íróként dolgozik. Gary nagy Linux-rajongó és vállalkozását a Linuxra és a nyílt forrású programokra alapozta. Szakterülete a reklámszövegírás és az üzleti kommunikáció.

Lapolvasás a SANE segítségével

Lássuk, hogyan vehetünk használatba egy lapolvasót Linux alatt, majd hasonlítsuk össze az elérhető alkalmazások tudását.

Néhány évvel ezelőtt a Linuxot használóknak gyakran gondot okozott a megvásárolt videoeszközök és az őket működtető XFree86 illesztőprogramok között fennálló bonyolult kapcsolatrendszer. Ha valaki meg akarta érteni egy új videokártya beállításának a módját, részletes ismereteket kellett szereznie az eszközről, például pontosan tudnia kellett a kártyán található lapkakészlet típusát. Bár ezek a gondok lassan eltűntek, ahogy a gyártók közvetlenül is támogatni kezdték az XFree86 fejlesztéseket, illetve saját XFree86 illesztőprogramok készítésébe fogtak, ám a linuxos képolvasás világa még mindig ebben a kezdetleges állapotban rekedt meg.

A vas

A lapolvasók általában háromféle módon kapcsolódhatnak a számítógéphez: párhuzamos kapun, SCSI-vezérlőkártyán vagy USB-kapun keresztül. Ma már szinte lehetetlen olyan lapolvasót találni, ami nem támogatja az USB-t, bár jelentős részük még a párhuzamos felületet is kezeli. A SCSI-felület támogatása – az USB javára – gyakorlatilag eltűnt.
A 2.4-es Linux-rendszermagok USB-támogatása a rendszermag USB-lapolvasó illesztőprogramjával vagy a `libusb` könyvtáron keresztül használható ki. A rendszermag képolvasó illesztőprogramjának jellemzői az alábbi paranccsal kérdezhetők le:

```
lsmod | grep scanner
```

Ha a parancs visszaad valamit, akkor a gépen van rendszermagszintű képolvasó illesztőprogram. Ellenkező esetben a képolvasó illesztőprogram az alábbi paranccsal tölthető be:

```
modprobe scanner
```

A parancs egyben az USB-magmodul betöltését is elindítja (ez `usbcore` néven szerepel az `lsmod` által adott listában), amennyiben ez korábban még nem történt meg. Ha a lapolvasót USB-felületen szeretnénk használni, a megfelelő USB `HCI` modult is be kell tölteni. USB 1.0-s eszközök esetén az `usb-uhci`, USB 1.1-es eszközöknél pedig az `usb-ohci` modulra van szükség. Az USB 2.0 alapú eszközöknél – kis sebességnél is – az `usb-ehci` végzi a vezérlést. Az én USB-s eszközöm USB 1.1 alapú, tehát nálam az `OHCI` változatot kell betölteni:

```
modprobe usb-ohci
```

Ha inkább a `libusb` könyvtárat akarjuk használni, az alábbi parancsot rendszergazdaként kiadva a lapolvasó modulokat el kell távolítani:

```
rmmod scanner
```

Noha egyes pletykák szerint a 2.6-os rendszermagokból el fog tűnni a képolvasó illesztőprogram, a jelenlegi 2.4-es rendszermagokban még megtalálható. A továbbiakban tehát feltételezem, hogy a rendszermag lapolvasó illesztőprogramját használjuk.
A lapolvasó alkalmazást egy parancsfájllal lehet elindítani, ami lefuttatja a megfelelő `modprobe` parancsokat, így minden esetben biztosak lehetünk abban, hogy a lapolvasó illesztőprogramok be lesznek töltve. Egy másik lehetőség a rendszer valamelyik indító parancsfájljának használata a lapolvasóhoz szükséges modulok betöltésére; Red Hat rendszerek esetében például a */etc/rc.local* fájl alkalmas erre.
A lapolvasó illesztőprogram betöltése után – ismét rendszergazdaként – az USB fájlrendszer befűzését kell elvégezni:

```
mount /proc/bus/usb
```

Ezt követően megjeleníthetjük az USB-buszon található eszközök listáját:

```
cat /proc/bus/usb/devices
```

A fenti parancs semmilyen kimenetet nem ad, ha a géphez nincs lapolvasó csatlakoztatva, vagy az USB-mag és a `HCI` illesztőprogram (vagyis az `uhci`, `ohci` vagy `ehci` valamelyike; a korábban leírtak szerint) nincs betöltve. A *devices* fájl részletes adatokat szolgáltat, de most a gyártó- és termékazonosítók a fontosak, például:

```
Vendor=04b8 ProdID=011d Rev= 1.00
```

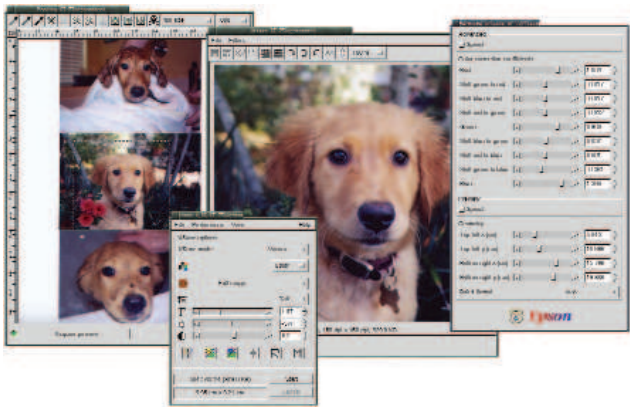
Ezeket az értékeket akár fel is lehet írni, mivel később, ha a SANE nem ismeri fel a lapolvasót, még szükség lehet rájuk. Ha biztosak akarunk lenni abban, hogy a SANE látja a lapolvasót, töltsük be újra a lapolvasó illesztőprogramot:

```
rmmod scanner
modprobe scanner vendor=0x4b8 product=0x011d
```

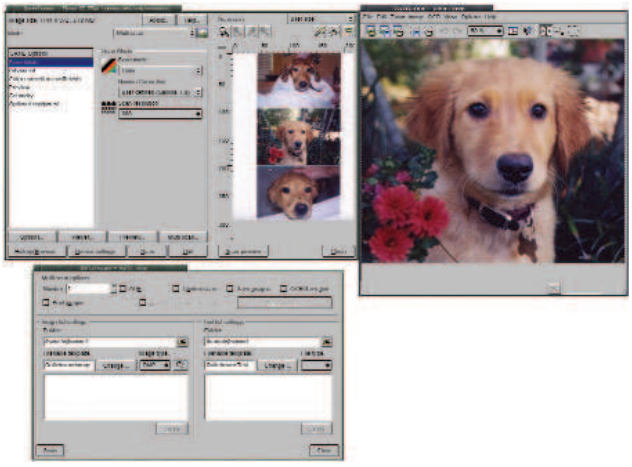
Ez alkalommal a lapolvasó illesztőprogram betöltésekor a gyártó- és termékazonosítót is megadtuk. Az azonosítók elé `0x` került – erre akkor van szükség, ha a `modprobe` parancsot így használjuk.
A sane-usb súgóoldal bővebb tájékoztatással is szolgál az USB-s lapolvasók beállításával kapcsolatban. Az általános USB-beállításokkal és a kipróbálásukkal kapcsolatban a Linux USB Project weblapján (➲ http://www.linux-usb.org) találhatók bővebb útmutatások.

Az eszközök

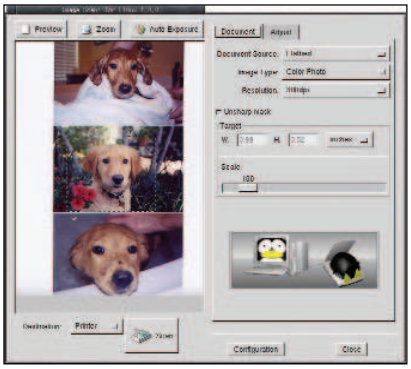
Most, hogy az alapvető eszközbeállítások megvannak, ellenőrizni kell, hogy a SANE képes-e az eszközök használatára. Cikkem írásakor a SANE 1.0.8-as változatát használtam. A SANE két részből áll: az illesztőprogram háttérrendszerből (back-end) és a felhasználói felületekből (front-end). A SANE jelenleg csak



1. kép Az XSane felület Epson Perfection 1260 lapolvasóval



2. kép A QuiteInsane felület Epson Perfection 1260 lapolvasóval



3. kép Az Image Scan! felület Epson Perfection 1260 lapolvasóval

a háttérrendszer illesztőprogramjait biztosítja, valamint néhány parancssori programot bocsát a rendelkezésünkre. Az X alapú, grafikus felületek – mint például az XSane és a QuiteInsane – külön tervezet keretében készülnek, és a SANE háttérrendszerre épülnek. Az újabb Linux-terjesztések már tartalmazzák a SANE háttérrendszer valamelyik változatát. Sajnos számos terjesztésben elég régi változatot találni. A SANE webhelyről (➲ http://www.mostang.com/sane) RPM és egyéb formátumokban érhetők el az időszerű változatok, Red Hat, Debian, Mandrake és Slackware terjesztésekhez.

Ha a SANE háttérrendszer telepítése megtörtént, akkor el kell végezni az illesztőprogramok beállítását. Az első buktató, hogy ellenőrizni kell, vajon a SANE megtalálja-e a lapolvasót. A SANE rendelkezik egy parancssori eszközzel, ez a `sane-find-scanner`, ami bármelyik SCSI-felületű lapolvasót felismeri, illetve a legtöbb USB-s készülékkel is boldogul. A segédprogram nemcsak rendszergazdaként, de egyszerű felhasználóként is futtatható, további kapcsolókra vagy átadott értékekre nincs szüksége. Kimenete néhány megjegyzést tartalmaz, illetve egy, az alábbihoz hasonló sort (USB-s lapolvasók esetében):

```
sane-find-scanner: found USB scanner
      (vendor = 0x04b8, product = 0x011d)
      at device /dev/usb/scanner0
```

Ez azt jelenti, hogy a SANE a */dev/usb/scanner0* eszközön keresztül látja a lapolvasót, ami jó dolog, de még el kell végezni az illesztőprogramok beállítását. Ha nem jelenik meg a fenti sor, vagyis a program nem találta meg a lapolvasót, akkor – ha a `sane-find-scannert` felhasználóként futtattuk – lehetséges, hogy módosítani kell az eszközfájlra vonatkozó jogosultságokat. Ennek szükségességéről a parancs rendszergazdaként való futtatásával győződhetünk meg. Ha a `sane-find-scanner` rendszergazdaként futtatva megtalálja a lapolvasót, akkor a jogosultságokkal van a gond. Ha te vagy az egyetlen felhasználó a számítógépen, a hiba könnyedén elhárítható:

```
chown tulajdonos.tulajdonos /dev/usb/scanner0
chmod 660 /dev/usb/scanner0
```

A fenti példában a tulajdonos a saját felhasználóazonosítód és csoportod. Ha más felhasználókkal kell osztozni a lapolvasón, akkor inkább hozzunk létre egy lapolvasó csoportot:

```
chgrp lapolvaso /dev/usb/scanner0
chmod 660 /dev/usb/scanner0
```

A fenti két parancsot rendszergazdaként kell futtatni.
A lapolvasó csoporthoz minden olyan felhasználót hozzá kell adni, aki használni szeretné a lapolvasót. Még egy megjegyzés az eszközfájllal kapcsolatban: nagy lehet a csábítás, hogy egy */dev/usb/scanner0* – */dev/scanner* közvetett hivatkozást hozzunk létre. Ne tegyük! A SCSI-háttérrendszer a */dev/scanner* eszköznevet használja, és ha ezzel az USB-s eszközre hivatkozunk, akkor megzavarjuk az USB-háttérrendszer működését.
Ha a SANE látja a lapolvasót, ideje kiválasztani a megfelelő háttérrendszer-illesztőprogramot. A dolog itt kezd rázós lenni. A legtöbb Epson lapolvasó az Epson háttérrendszert használja. Az Epson Perfection 1260 – jó áron kapható, a legtöbb üzletben beszerezhető modell – viszont a Plustek illesztőprogrammal működik. A legtöbb lapolvasó esetében elég jól meg lehet tippelni a megfelelő választást, ha előbb a SANE weboldalán áttanulmányozzuk a támogatott eszközök listáját. Ha így sem érünk el sikert, akkor a gyártó és a termék azonosítója alapján a Linux USB Project webhelyén található támogatott eszközök listájával juthatunk előbbre.

A SANE beállításai

A megfelelő illesztőprogram kiválasztása után meg kell adni a háttérrendszer beállításait. Ha RPM-ből telepítettük a SANE-t, vagy forrásból építve az alapértelmezett beállításokat választottunk, akkor a SANE beállítóállományai a */etc/sane.d* könyvtárba kerülnek. A fő beállítófájl a ***dll.conf*** , ebből tudja a SANE, hogy mely illesztőprogramokat kell használnia. Alapállapotban

elég sok illesztőprogram engedélyezett. Ha nincs mindre szükség, akkor az adott lapolvasóhoz tartozót kell csak engedélyezni. Példámban csak az Epson illesztőprogram van engedélyezve, mivel a Plustek illesztőprogram helyett az Epson KOWA háttérrendszerre volt szükség.
Nem minden háttérrendszer támogatja az összes lapolvasótípust. Mindegy, hogy milyen kapcsolattal használjuk a lapolvasót, az összes háttérrendszer-beállítóállományban be kell állítani a lapolvasó elérésére használt eszköz nevét. Emlékeztetőül: az eszköz fájlnevét a `sane-find-scanner` segédprogrammal találtuk meg. Sajnos a beállítófájlokban való megadás formátuma háttérrendszertől függően változik. A Plustek háttérrendszer (a SANE alapértelmezett háttérrendszere az Epson Perfection 1260 támogatásához) külön részeket használ az USB-s és a párhuzamos kapura csatlakozó lapolvasókhoz. Ebben a beállítófájlban a `device` kulcsszó után kell írni az eszközfájl nevét, így adható meg az eszközfájl; lásd a példát:

```
device /dev/usb/scanner0
```

A bejegyzést a Plustek illesztőprogram beállítóállományának megfelelő részében kell elhelyezni. Az Epson Perfection 1260 lapolvasóhoz használt Epson KOWA háttérrendszer Epson beállítóállományában például az `usb` kulcsszót kell használni, ezt követi az eszközfájl neve:

```
usb /dev/usb/scanner0
```

Az Epson és a Plustek beállítófájl egyaránt tartalmaz megjegyzéseket, ezek alapján könnyebben megadhatók a beállítások, emellett minden háttérrendszerhez saját súgóoldal tartozik, amelyben további tájékoztatást is találni. A rengeteg beállítás között minden esetben az egyetlen igazán fontos az eszközfájl neve.
A SANE beállításainak működőképességét az alábbi paranccsal lehet ellenőrizni:

```
scanimage -T
```

Ha a próba sikertelen, akkor újra ellenőrizni kell, hogy a megfelelő USB-modulok telepítve vannak-e, a `sane-find-scanner` megtalálja-e az eszközt, illetve pontosan szerepel-e az eszközfájl neve a SANE háttérrendszer beállítóállományában. A `scanimage` program `help` kapcsolójával további adatokat tudhatunk meg a lapolvasó képességeiről:

```
scanimage --help
```

Grafikus felületek

A gépet beállítottuk, használatra készen áll. Ami még hiányzik, az egy könnyen használható grafikus felület, amivel megtekinthetjük a beolvasások előnézetét, kiválaszthatjuk az előnézetek beolvasandó területeit, illetve szükség szerint színkorrekciót végezhetünk vagy megadhatjuk a kívánt minőséget és a felbontást. Ha a beolvasott képet adott esetben a Gimp alatt további feldolgozásnak szeretnénk alávetni, erre is valamilyen módot kell találni.
A SANE-hez jelenleg három ingyenes grafikus felület érhető el, illetve létezik egy korlátozott használatú (shareware) alkalmazás is, ami független a SANE-től. Mielőtt összehasonlítanánk a tudásukat és a munkájuk minőségét, röviden ismerkedjünk meg velük.

XSane

A tervezet a SANE Projecttel együtt fejlődött. Felhasználói felülete GTK+ alapú, és tartalmaz egy Gimp beépülő modult is, amivel a Gimp ***File Acquire*** menüpontján keresztül közvetlenül is beolvashatunk képeket.
A Gimp beépülő modulként futtatva a ***Képnézegető*** (Viewer) ablakot nem használja, a beolvasott kép közvetlenül a Gimp alkalmazásba, egy ***Vászon*** (Canvas) nevű ablakba kerül. A leírását érdemes figyelmesen elolvasni, hogy a legtöbbet hozhassuk ki a programból – külön ajánlom a lapolvasási tanácsokat tartalmazó weboldalakra vezető hivatkozásokat.
Az ***Előnézet*** (Preview) ablakban az olvasási területek önműködően és kézzel is kijelölhetők. Előnézetben fekete, szürke és fehér pontok is megadhatók, még a teljes méretű kép beolvasása előtt. A ***Képnézegető*** ablak korlátozott szerkesztési lehetőségeket nyújt.

QuiteInsane

Az XSanehez hasonlóan a QuiteInsane is rendelkezik Gimp beépülő modullal. A modul a fejlesztés korai szakaszában van, és nem olyan üzembiztos, mint az XSane modulja. Ettől eltekintve a QuiteInsane nagyjából hasonló tudással bír, mint az XSane, illetve néhány extrát is nyújt. A QuiteInsane képnézegetője segítségével a felhasználó kiválaszthatja, hogy a kép mely részeivel szeretné folytatni a munkát, és közvetlenül innen lehet nyomtatni is.
A QuiteInsane beépül a KDE munkaasztalba, az Image Viewerből (Képnézegetőből) fogd és vidd módszerrel a képek más alkalmazásokba is áthúzhatók. Lehet, hogy a más alkalmazásokkal való együttműködésre nem mindig van szükség, ám az átlagos felhasználók számára ez fontos kényelmi szolgáltatás.

Image Scan!

Az egyetlen olyan lapolvasókat gyártó cég, ami tevőlegesen támogatja a SANE Projectet, az Epson. Az Epson KOWA Corporation Image Scan! névvel saját felhasználói felületet készített, ami továbbfejlesztett háttérrendszerével az összes jelenlegi Epson lapolvasót támogatja. Előnye, hogy egyenesen a gyártótól érkező, frissített illesztőprogramokat tartalmaz az üzletekben megvásárolható készülékekhez.
Felülete ugyan kevésbé kifinomult, mint az XSane vagy a QuiteInsane, szolgáltatáskészlete is szűkösebb, ám egyben tisztább, áttekinthetőbb. Beépített képnézegetővel sem rendelkezik, így a Gimpet használhatjuk a képek szerkesztésére, vagy azonnal fájlba menthetjük, esetleg kinyomtathatjuk őket. Beépített súgóval nem rendelkezik, online leírása is szűkszavú.

VueScan

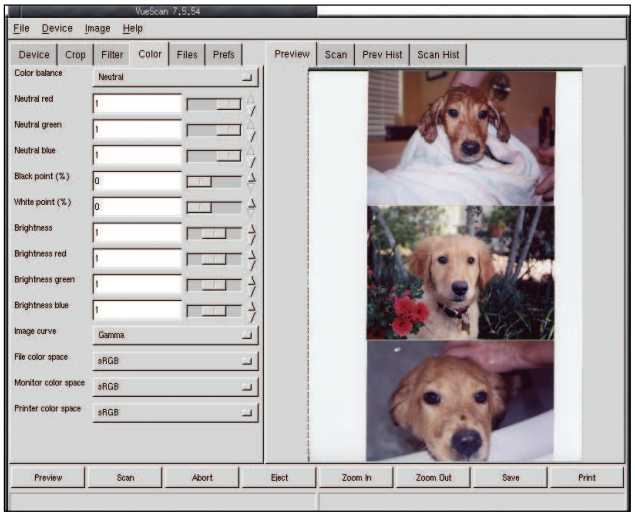
Az XSane, a QuiteInsane és az Image Scan! a SANE háttérrendszer lapolvasó illesztőprogramjaihoz szolgáltat felhasználói felületet. Létezik egy olyan alkalmazás, ami nem a SANE háttérrendszerre támaszkodik: ez a Hamrick Software VueScan terméke. Az alkalmazás korlátozottan használható, és számos lapolvasóhoz saját illesztőprogrammal rendelkezik. A VueScan jó néhány olyan szolgáltatást nyújt, amelyet a SANE-hez készült felületekben nem találunk meg, ilyen például az eszközök kalibrálása, a fókusz és a megvilágítás szabályozása. Képszerkesztésre nem alkalmas, Gimp beépülő modulja nincs. A beolvasott képeket először fájlba kell menteni, majd a megfelelő képszerkesztőben megnyitni.

Lapolvasó felhasználói felületek összehasonlítása

		Xsane	QuiteInsane	Image Scan!	VueScan
	Háttérrendszer	**SANE**	**SANE**	**SANE/Epson KOWA (csak Epson lapolvasóknál)**	**Beépített (számos USB, SCSI és párhuzamos felületű modellt támogat)**
Felhasználói felület	Különálló	GTK+	Qt	GTK+	GTK+
	Gimp beépülő modul	GTK+	GTK+ (Qt is szükséges)	GTK+	Nem
	Egyéb stílusok	–	Windows, Motif, Platinum, SGI és CDE	–	–
	Eszköztippek	Igen	Igen	Nem	Nem
	Előre beállított adathordozó-típusok	Igen	Nem	Igen	Igen
	Súgó, leírás	Külső böngésző	Belső böngésző	Korlátozott súgóoldal	Netscape böngésző
	Vonalrajz, szürkeárnyalatos, színes	Háttérrendszertől függő	Háttérrendszertől függő	Lapolvasótól függő	Lapolvasótól függő
	Állítható beolvasási felbontás	Igen	Igen	Igen	Igen
	Állítható előnézet felbontás	Igen	Igen	Nem	Igen
	Önműködő fájlnév-előállítás	Igen	Igen	Nem	Igen
	A felhasználói szerződés jellege	GPL, nyílt forrás	GPL, nyílt forrás	Nyílt forrás – GPL, LGPL és Epson KOWA nyilvános szerződés	Korlátozott használat (shareware)
A mentéshez használható formátumok	PNG	Igen	Igen	Igen	Nem
	JPEG	Igen	Igen	Nem	Igen
	TIFF	Igen	Igen	Nem	Igen
	PNB/PBM család	Igen	Igen	Igen	Nem
	XPM/XBM	Nem	Igen	Nem	Nem
	Egyéb	PS, RAW	BMP	Nem	RAW
Beolvasás célja	Gimp beépülő modul	Igen	Igen	Igen	Nem
	Belső képnézegető	Igen	Igen	Nem	Igen, külső adható meg
	Nyomtatás	Igen (másolatként)	Igen	Igen	Igen
	Fájl	Igen (másolatként)	Igen	Igen	Igen
	Elektronikus levél	Igen	Nem	Nem	Nem
	Fax	Igen	Nem	Nem	Nem
	Karakterfelismerés	Igen	Igen	Nem	Nem
	Egyéb	Nem	Több beolvasás	Nem	Nem
Színkorrekció	RBG gamma/intenzitás szintek	Igen	Igen	Igen	Igen
	Színeltolás	Igen	Igen	Nem	Nem
	Fehér/fekete/szürke pontok megadása	Igen	Nem	Nem	Igen
	Fényerő/képélesség	Igen, olvasás előtt	Igen, olvasás után	Nem	Igen, több csatorna
	Egyéb	Nem	Nem	Fényfoltok, árnyékok, küszöbérték	Előzetes színbeállítás, fájl, monitor és nyomtató-színkezelés
A képnézegető szolgáltatásai	Kiválasztható területek	Nem	Igen	Nincs képnézegető ablak	Nem
	Nyomtatás a képnézegetőből	Nem	Igen	–	Igen
	Művelet visszavonása/újra elvégzése	Nem	Igen	–	Nem
	Szűrők	Szemcsék eltávolítása,elkenés	Szemcsék eltávolítása, elkenés, invertálás, normalizálás, olajfestmény, poszter, élkiemelés	Elmosás maszk az előnézet párbeszédablakban	Színek helyreállítása, fakulás helyreállítása, szemcsésség csökkentése, élkiemelés
	Forgatás, átalakítások	Forgatás, tükrözés	Forgatás, méretezés, nyírás	-	Balra, jobbra, átfordítás
Előnézeti szolgáltatások	A beolvasott terület önműködő kiválasztása	Igen	Állítható	Nem	Igen
	A látható terület önműködő kiválasztása	Igen	Igen	Csak kézzel választható	Igen
	Nagyítás előnézet	Igen	Igen	A kiválasztott terület újraolvasásával	Igen
	Előre megadott beolvasási méretek	Igen, amerikai és európai	Igen, európai	Nem	Igen, több típus
	Forgatás, átalakítások	Nem	Nem	Nem	Balra, jobbra előnézetből vagy előolvasásból, átfordítás, tükrözés
	Az előnézet frissítése	Igen	Igen	Igen	Nem
	Folyamatjelző	Nem	Nem	Igen	Nem

Az alkalmazások összehasonlítása

A négy felhasználói felület közül csak egy kötődik valamely gyártóhoz: az Image Scan! csak Epson lapolvasókkal használható. A másik három különféle gyártókat támogat. Mindegyik felület a beolvasás előtt, illetve után képes különféle gamma- és színcsatorna-korrekciók elvégzésére. Az XSane, a QuiteInsane



4. kép A VueScan-felület Epson Perfection 1260 lapolvasóval

5. kép Összehasonlítás: bal – SANE; jobb – VueScan. A SANE változatnál a JPEG-minőség 100-ra volt állítva

és a VueScan beépített, bizonyos szerkesztési műveletek elvégzésére is alkalmas képnézegetővel rendelkezik. Az Image Scan! csak fájlba vagy közvetlenül a Gimpbe tud olvasni.
A legnagyobb különbség az alkalmazások között abban mutatkozik, hogy a beolvasott képeket hova tudják átadni, illetve hogyan lehet őket kinyomtatni. Az XSane támogatja a legtöbb célt, köztük az elektronikus levelet és a faxot. Az optikai karakterfelismerést (OCR) az XSane és a QuiteInsane egyaránt támogatja a külső `gocr` program segítségével.
Az XSane, a QuiteInsane és az Image Scan! folyamatosan frissíti az előnézetet, miközben az előnézeti beolvasást végzik. A felhasználó tehát ezeknél figyelemmel követheti a beolvasás menetét. Ugyanez vonatkozik a végleges beolvasásokra.
A VueScan nem végez folyamatos frissítést.
A nyomtatásban úgyszintén hatalmas különbségek mutatkoznak. A QuiteInsane az egyetlen, amiből – különféle beállításokkal – nyomtatni is lehet. Az XSane ugyan képes közvetlenül nyomtatóra beolvasni, de a beépített képnézegetőből nem lehet nyomtatni. A VueScan képnézegetőjéből ugyan tudunk nyomtatni, de a felületet szolgáltatásai szegényesek.
Az XSane és a QuiteInsane leírása minden részletre kiterjedő, igényes. A VueScanhez is tartozik HTML formátumú leírás, de kevésbé részletes. Az Image Scan! csak egy csekély méretű súgóoldallal rendelkezik.
A nyílt forrású eszközök Gimp beépülő modullal is rendelkeznek, a VueScan viszont nem. Az XSane beépülő modulja a legmegbízhatóbb. A QuiteInsane modulja ugyan sok szolgáltatást nyújt, ám a fejlesztés korai szakaszában van, és nem olyan üzembiztos, mint amennyire azt a felhasználók nagy része valószínűleg elvárná. Az Image Scan! modulja ugyanazokkal a képességekkel rendelkezik, mint maga az alkalmazás, az XSane és a QuiteInsane esetében azonban a Gimp-modul és

6. kép Ráközelítve a SANE- (Epson) és VueScan-képekre felfedezhetők a minőségbeli eltérések

az alkalmazás tudása némileg eltér egymástól.
Táblázatunkban a négy felület összehasonlítása található.
A táblázat jó alapul szolgálhat a megfelelő felület kiválasztásához, de természetesen mindegyikkel érdemes legalább egy próbát tenni.

Minőségi különbségek

Az XSane és a QuiteInsane esetében a beolvasások eredménye lényegében megegyező, hiszen ugyanarra a közös SANE háttérrendszerre támaszkodnak. Az Image Scan! illesztőprogramja az Epson windowsos illesztőprogramjával azonos minőségre képes. A VueScan saját illesztőprogramot használ. Az *5. képen* látható két kép alapján össze lehet hasonlítani az Epson és a VueScan illesztőprogramját. A *6. képen* a szemek környékét nagyítottam ki. A SANE változatnál finomabb az átmenet a képpontok között, míg a VueScan a bal szemben látható tükröződéseknél részletgazdagabb.

Összegzés

Mindegyik felületnek megvannak a maga előnyei és hátrányai.
Az XSane a filmről – Agfa, Fuji és Kodak negatívokról – végzett olvasást is támogatja, használata egyszerű.
Az Image Scan! inkább az átlagfelhasználónak készült. Érdekessége, hogy illesztőprogramja közvetlenül az

Epsontól származik, és támogatja az általánosan hozzáférhető lapolvasókat.
A VueScan segítségével gyorsabban ki lehet nagyítani az előnézet egy-egy részletét, ugyanis a képet a memóriában tárolja. A VueScan emiatt a SANE alapú megoldásoknál jobban használható, ha nagy számú képet kell beolvasni.
Csak a VueScan támogat többféle színrendszert: sRGB, PAL, NTSC, CIE, Apple, Adobe és továbbiak. Mindemellett ez volt az egyetlen alkalmazás, ami a próbák során összeomlott.
A QuiteInsane számos olyan szolgáltatást nyújt, amiket a többi felületből hiányolhatunk. Például a menüsorok elmozdíthatók, ezt a lehetőséget a Qt használata biztosítja. Rendelkezik egy, a felhasználó által módosítható színgörbével is, ami a Gimp *Curves* eszközéhez hasonlóan a vörös, zöld és kék csatorna módosítására használható. A *Viewer* ablakából közvetlenül nyomtathatunk, megadva a kívánt méretet, a margókat, a képfelbontást és az oldalméretet.
Nincsenek az adathordozó típusától függő, előre megadott színkorrekciós beállítások, és a beépített lapméretek sem tartalmazzák az amerikai szabványméreteket a levelekhez és az iratokhoz. A hiányzó szolgáltatások ellenére talán kijelenthetjük, hogy QuiteInsane a legjobb választás.
Akár profi művészről, akár alkalmi fotósról legyen szó, megfelelő linuxos lapolvasási megoldást találhat. Mindegyik felület más valamiben, mint a többi, a háttérrendszer pedig nagyszámú illesztőprogramot biztosít, egyes gyártók pedig – mint az Epson – külön változatot fejlesztenek saját termékeik támogatására. Segítségükkel Linux alatt is ugyanolyan jól lehet képeket beolvasni, mint bármilyen más asztali operációs rendszer alatt.

KAPCSOLÓDÓ CÍMEK

Epson KOWA
➲ http://www.epkowa.co.jp/english/linux_e/lsd_e.html
Image Scan!
➲ http://www.epkowa.co.jp/english/index_e.html
Linux USB Project ➲ http://www.linux-usb.org
QuiteInsane
➲ http://quiteinsane.sourceforge.net/index.shtml
SANE ➲ http://www.panda.mostang.com/sane
VueScan➲ http://www.hamrick.com
XSane ➲ http://www.xsane.org

Linux Journal 2003. április, 108. szám

Michael J. Hammel (mjhammel@graphics-muse.org)
Író, grafikusművész és programfejlesztő, jelenleg a texasi Houstonban dolgozik. Felszólalt az ALS, a LinuxWorld és az SXSW konferenciákon, illetve elnöke volt egy coloradói Linux-konferenciának. Weboldala, a The Graphics Muse (➲ http://www.graphics-muse.com) grafikusoknak és fejlesztőknek egyaránt fontos forrásokat biztosít Linux alá.

Az rsync (1. rész)

Az rsync csupán az eltérő állományrészeket továbbítja a hálózati csomópontok között.

Az *Andrew Tridgell* által készített rsync nagyszerű fájltovábbító eszköz, és ugyan nem rendelkezik beépített titkosítási lehetőséggel, ám könnyen burkolható (alagúton keresztül is használható) más titkosító eszközökkel, mint az SSH vagy az Stunnel. Amiben az rsync (amely az scp-hez hasonlóan rcp alapú) különbözik a többi eszköztől, az le- és feltöltési képessége.
Ha például frissíteni szeretnéd egy 10 MB-os fájlnak a számítógépeden található másolatát, és az újabb változat, ami egy távoli kiszolgálón található, csak három helyen különbözik a régitől, egy összesen 150 KB nagyságú szakaszon, akkor az rsync csak ezt az eltérő, nagyjából 150 KB méretű részt fogja letölteni, és nem az egész állományt. Ezt a lehetőséget az Andrew Tridgell és *Paul Mackerras* által kitalált rsync algoritmus biztosítja, ami pillanatok alatt gördülő ellenőrzőösszegeket (rolling chechksums) hoz létre mindkét fájlról, összehasonlítja őket, és így határozza meg, hogy az új fájl mely részeit kell letöltenie és a régi állomány szakaszait velük lecserélnie.
Mivel a hálózat így jóval hatékonyabban használható, az rsync különösen lassabb kapcsolatoknál hasznos. Természetesen semmilyen teljesítménybeli előnye nincs az rcp-hez képest abban az esetben, ha az átvitelben részt vevő felek valamelyike számára a fájl teljesen új. A különbségi másoláshoz értelemszerűen két összehasonlítandó fájlra van szükség.
Összefoglalva: az rsync messze a legintelligensebb fájlátviteli eszköz mindennapi használatra, amit érdemes biztonságos kapcsolatok kezelésére is képessé tenni, és ez esetben valóban érdemes megküzdeni e kérdéssel. Írásomban a továbbiakban az rsync biztonságos használatát ismertetem.

Az rsync beszerzése, lefordítása és telepítése

Mivel Andrew, az rsync eredeti vezető fejlesztője a Samba projektben is fontos szerepet tölt be, az rsync honlapja a hivatalos Samba webhely alatt, a ➲ http://rsync.samba.org címen található. Ez az rsync elsődleges beszerzési helye.
A ➲ http://rsync.samba.org/resources.html címen lévő „resources” (források) oldal további kiváló, de más webhelyeken található leírásokra is tartalmaz hivatkozásokat.
A legújabb rsync-forráskód a ➲ http://rsync.samba.org/ftp/rsync címen érhető el, futtatható csomagokat Debian, LinuxPPC és Red Hat Linux alá az ➲ http://rsync.samba.org/ftp/rsync/binaries címen lehet találni.
Az rsync ma már elfogadott linuxos segédeszköz, így minden népszerű terjesztésben megtalálható – lehet, hogy elég elővenned rendszered telepítő CD-it, és megtalálod a saját változatodhoz illő rsync-csomagot.
Az rsync 2.5.4-esnél korábbi változataiba beépített zlib sajnos biztonsági hiányosságokat tartalmaz. Ezek a hiányosságok a rendszeredben lévő megosztott zlib-könyvtárak változatától függetlenül fennállnak. Magában a 2.5.4-es változatban is akad egy bosszantó hiba, emiatt az rsync időnként – szükségtelenül – teljes állományokat másol át. Javaslom tehát, hogy ne használj 2.5.5-nél régebbi rsync-változatot.
Szerencsére az rsync fordítása forrásból gyors és könnyű feladat.

Az rsync futtatása SSH felett

Miután az rsync felkerült a gépre, sokféleképpen lehet használni. Az első és legalapvetőbb mód az rcp használata az átvitelekhez; ilyenkor azokon a gépeken, amelyekhez csatlakozni szeretnél, az *inetd.conf*-ban engedélyezni kell a héjszolgáltatást (például *in.rshd*). Ezt ne használd! A biztonságos héjszolgáltatást (Secure Shell) pontosan azért alkották meg, mert az „r”-szolgáltatások, mint az rcp, az rsh és az rlogin, tökéletesen nélkülözték az erős hitelesítés bármiféle támogatását, ennek következtében számos sikeres támadás kiindulópontját jelentették az elmúlt évek során.
Nem is fogom tehát leírni, hogy rcp alapon hogyan használható az rsync. Ha – például egy megbízható hálózat állomásai között – mégis ki szeretnéd ezt a módot próbálni, az rsync és az *in.rshd* leírásának vonatkozó részeiben bőséges útmutatót találsz.
Az rcp alapú módnál messze biztonságosabb az rsyncet biztonságos héjszolgáltatás felett használni. Ilyenkor a távoli állomásnak `sshd`-t kell futtatnia, illetve az `rsync` parancsnak mindkét félnél az alapértelmezett elérési útvonalon keresztül kell elérhetőnek lennie. Ha még nincs fent a gépeden az `sshd`, elsőként kezdd a telepítésével.
Tegyük fel, hogy két – egy helyi és egy távoli – gépről van szó, és a helyi gépen található *cuccok.tgz* fájlt szeretnéd átmásolni a távoli gép */home/helyi.mentes* könyvtárába, ami emlékeid szerint már tartalmazza ugyanennek az állománynak egy korábbi változatát. Feltéve, hogy mindkét gépen yodeldiva a felhasználóneved, az átvitelt így indíthatod el:

```
yodeldiva@helyi:~> rsync -vv -e ssh
➥./cuccok.tgz tÆvoli:~
```

Elemezzük a fenti sort. Az rsync egyetlen – *rsync* nevű – futtatható fájlból áll, mind az ügyfél, mind kiszolgáló oldalán ezt kell használni, akár démonként. A fenti példában a program mindkét oldalon elérhető, de egyiken sem fut démonként.
A távoli gépen az `sshd` feladata a bejövő kapcsolatok fogadása. Példánkban az első rsyncnek átadott kapcsoló a `-vv`, ami a Unix világában a very verbose (nagyon bőbeszédű) általánosnak mondható rövidítése. Nem muszáj használni, de érdekes adatokkal szolgálhat. A második kapcsoló a `-e`, amivel lecserélhetjük az rsync által alapértelmezettként használt távoli fájlmásoló programot, az rcp-t. Mivel az rcp az alapértelmezett, és az rcp-n és az SSH-n kívül nincs más választási lehetőség, a `-e` a gyakorlatban az SSH használatára utasít.
Talán meglepő, de a `-e scp` parancs nem működik, mivel az adatok másolásának megkezdése előtt az rsyncnek SSH-n keresztül ki kell adnia egy parancsot a távoli gépnek, a távoli állomány gördülő ellenőrzőösszegének kiszámítására és átküldésére utasítva őt. Vagyis az rsync az iménti művelet elvégzéséhez az `ssh` parancs teljes szolgáltatáskészletét igényli, tehát a `-e` kapcsoló használatakor inkább az `ssh`-t írd mögé, és ne az `scp`-t.
A kapcsolók után következik a helyi és a távoli fájl neve. Ezek megadása nagyon hasonló az rcp és az scp esetében megismerthez. Ha valamelyik fájlnév elé közvetlenül egy kettőspontot írsz,

A rsyncd.conf mintafájl

```
# kizÆr lag Ætfog  hat k rrel megadhat
# beÆll tÆsok
syslog facility = helyi5

# Ætfog  jellegß, de a moduloknÆl is
# megadhat  beÆll tÆsok
use chroot = yes
uid = nobody
gid = nobody
max connections = 20
timeout = 600
read only = yes

# pØldamodul:
[public]:
   path = /home/public_rsync
   comment = megjegyzØs sz vege
   hosts allow = helyi.valami.org, 10.18.3.12
   ignore nonreadable = yes
   refuse options = checksum
   dont compress = *
```

az rsync a kettőspont előtti részt a távoli gép neveként fogja kezelni. Ha a távoli gépen használandó felhasználóneved nem egyezik meg a helyi gépen használttal, akkor a távoli gép neve elé egy @ karaktert kell írnod, és ez elé kell begépelned a távoli felhasználónevedet. Az rsync fájlnév írásmódja tehát így alakul:

```
[[felhasznÆl nØv@]gØpnØv:]/a/fÆjl/elØrØsi/œtja
```

Legalább két fájlnevet kell megadni. A jobb szélső lesz a célfájl vagy célelérési út, a többiek pedig a forrásfájlok. A kettőből lehet távoli fájl, de akár mindkettő lehet helyi (ekkor nem kell kettőspontot használni), így helyi különbségi másolást is lehet végezni – ez különösen akkor jön jól, ha két helyi lemez vagy lemezrész között szeretnél biztonsági mentést végezni.
A fent megadott forrásfájl a *./cuccok.tgz*, ez egy normál helyi fájl elérési útja; a célfájl pedig `tÆvoli:~`, amit a rendszer így értelmez: a */home* könyvtáram a távoli kiszolgálón. Ha a távoli gépen más a felhasználóneved, mint a helyin, tehát mondjuk yodeldiva helyett yodeldavi, akkor a cél `yodeldavi@tÆvoli:~` lesz.

Az rsync kiszolgáló üzembe helyezése

Az rsync-SSH párosítás a legegyszerűbb módja az rsync hitelesített felhasználókkal való biztonságos használatának, és ez nemcsak megköveteli a valódi felhasználói fiókok alkalmazását, de védi is az adataikat. Mint az „SFTP és SSH” című részben említettem, az SSH nem egykönnyen vehető rá a névtelen hozzáférés támogatására. Mit kell tehát tenned, ha nyilvános, rsync alapú átviteleket támogató fájlkiszolgálót szeretnél létrehozni?
A megoldás egyszerű: hozz létre egy */etc/rsyncd.conf* állományt, az rsyncet pedig a `--daemon` kapcsolóval futtasd (pl.: `rsync --daemon`). Az ördög a részletekben rejlik. A */etc/rsyncd.conf* tartalmát roppant körültekintően kell összeállítanod, ha a kiszolgálót az Internetre, vagy más, nem megbízható hálózatra is csatlakoztatni szeretnéd. Lássuk a tudnivalókat!
Az *rsyncd.conf* írásmódja egyszerű, az átfogó hatókörű (global) beállítások behúzás nélkül kerülnek az elejére. A modulokat, amelyek kifejezetten a fájlrendszer valamely elérési útjára vonatkozó beállítások csoportjai, szögletes zárójelekkel közrefogott nevük vezeti be, ezt a megfelelő beállítások követik.
A beállításorok mindegyike tartalmazza magának a beállításnak a nevét, ezután egy egyenlőségjel, majd a kívánt érték vagy értékek állnak. Ha az adott beállítás logikai jellegű, a megengedett értékek a `yes` és a `no` (ne hagyd magad megtéveszteni az rsyncd.conf(5) súgóoldala által, ami egyes esetekben ***true*** és ***false*** értékekre utal). Ha egy beállítás több értéket is kaphat, ezeket vesszővel és szóközzel kell elválasztani egymástól, mint például: `ØrtØk1, ØrtØk2`.
Az első kódrészlet egy *rsyncd.conf* mintafájlból származik, ami néhány, különösen biztonsági szempontból fontos beállítást szemléltet. Bár csupán bemutató célra készítettem, ez is valódi beállítófájl és az írásmódja is teljesnek tekinthető. Boncoljuk tehát fel!
A korábbiak szerint az átfogó hatókörű beállítások felülre kerültek. Ez az első beállításcsoport kizárólag olyan beállításokat tartalmaz, amelyek csak átfogó hatókörűek lehetnek: `syslog facility`, `motd file`, `log file`, `pid file` és `socket` – ezeket a beállításokat csak átfogó jelleggel lehet megadni, a moduloknál nem. Ezek közül csak a `syslog facility` beállításnak vannak közvetlen biztonsági vonatkozásai.
A ProFTPD ***SyslogFacility*** beállításához hasonlóan az rsync `syslog facility` beállítása is arra szolgál, hogy megadjuk a naplózásra az rsync által használandó eszközt, ha nem akarjuk, hogy démont használjon (alapértelmezett). A kizárólag átfogó hatókörrel használható beállítások részletes leírását lásd az rsyncd.conf(5) súgóoldalon; ezekkel itt nem foglalkoznék, mivel a rendszer biztonságát közvetlenül nem befolyásolják, és a legtöbb esetben alapértelmezett értékük is bátran használható.
A többi ***rsyncd.conf***-beállítás átfogó jelleggel, modulokban vagy mindkét helyen tetszés szerint használható. Ha egy beállítás az átfogó részben és valamelyik modulban is megjelenik, akkor a modult érintő átvitelek esetében a modulbeli érték felülbírálja az átfogót. Általában elmondhatjuk, hogy az átfogó beállítások értékei felülbírálják az alapértelmezetteket, a modulok értékei pedig az alapértelmezett és az átfogó értékeket egyaránt.
A beállítások második csoportja a moduloknál használható.

- `use chroot = yes`: ha a `chroot` értéke `yes`, az rsync a fájlátvitel megkezdése előtt `chroot` műveletet hajt végre a modul elérési útjába, így megakadályozhatók bizonyos visszaélések és támadások, vagy legalábbis megnehezíthetők. A dolog azzal jár, hogy az `rsync --daemon` parancsot a rendszergazdának kell kiadnia, de azt az időt, amit az rsync rendszergazdai módban tölt, az `uid` és `gid` beállítások módosításával is csökkentheted. Alapértelmezett értéke `yes`.
- `uid = nobody`: az `uid` beállítás révén adható meg, hogy az rsync milyen felhasználói jogosultságokkal fusson a fájlátvitelek során, illetve azt is befolyásolja, hogy az ügyfelek kiszolgálásakor az rsync milyen jogosultságokkal próbálja írni vagy olvasni az állományokat. Értéke felhasználónév vagy számjegyekkel megadott azonosító is lehet. Alapértelmezett értéke `-2`, ami sok rendszernél a ***nobody*** felhasználó. Az enyémnél ez nem így van, ezért közvetlen formában adtam meg az `uid` értékét.
- `gid = nobody`: a `gid` beállítás szabja meg, hogy az rsync milyen csoport jogosultságaival fusson a fájlátvitelek során, illetve az `uid` beállítással együtt arra is hatással van, hogy az rsync milyen jogokkal próbálja írni vagy olvasni az állományokat az ügyfelek kiszolgálásakor. Felhasználónév és numerikus azonosító is megadható, alapértelmezett értéke `-2` (a legtöbb rendszeren a ***nobody*** azonosítója).

- `max connections = 20`: az egy időben fennálló kapcsolatok számát korlátozza egy-egy modulra vonatkozóan (bár átfogó beállítás, nem az összes modulnál fennálló kapcsolatok összegére vonatkozik). Ha átfogó hatókörrel adjuk meg, az értéke minden olyan modulra érvényes lesz, ami nem rendelkezik saját kapcsolatszám-korláttal. Alapértelmezett értéke nulla, ami azt jelenti, hogy az egy időben létrehozható kapcsolatok száma nincs korlátozva. Nem javaslom, hogy nullán hagyd az értékét, mivel ezzel megkönnyítenéd a szolgáltatásmegtagadási (DoS-) támadások indítását.
- `timeout = 600`: alapértéke szintén nulla, ami lényegében a „nincs korlát” beállítást jelenti. Mivel a `timeout` adja meg, hogy az rsync – másodpercekben mérve – milyen hosszan vár a tétlen kapcsolatok újraaktiválódására, helytelen értéke szintén DoS-támadásokra adhat módot, és átfogó jelleggel kell megadni (illetve modulonkénti jelleggel is megadható, ha az adott modulhoz valamiért eltérő értéket kell használni).
- `read only = yes`: az utolsó átfogó jellegű beállítás a `read-only`, ami a kérdéses modul csak olvasható jellegét szabja meg; csak olvasható könyvtár alá nem lehet fájlokat vagy könyvtárakat feltölteni, mindössze a meglévőeket letölteni. Alapértelmezett értéke `yes`.

A beállítások harmadik csoportja a `[public]` modullal kapcsolatos. Mint látható, ezek a sorok már be vannak húzva. Amikor az rsync lefelé haladva feldolgozza az ***rsyncd.conf*** állományt, minden valamilyen modulnév alatti beállítást az adott modulhoz tartozóként kezel, amíg egy másik – szögletes zárójelekkel jelölt – modulnévhez nem érkezik, vagy el nem éri a fájl végét. Tekintsük át a `[public]` modulhoz tartozó beállításokat!

- `[public]`: ez a modul neve. Más kapcsolók, átadott értékek nem tartoznak hozzá, egyedül a modul nevét kell megadni, ami ebben az esetben `public`.
- `path = /home/public_rsync`: a `path` beállítás minden modulnál kötelező, ugyanis ez adja meg, hogy a modul melyik könyvtár állományaihoz biztosít olvasási és írási lehetőséget. Ha az átfogó beállítások között a `use_chroot yes` értéket kap, akkor az rsync ebbe a könyvtárba fog `chroot` műveletet végrehajtani a fájlátvitelek megkezdése előtt.
- `comment = megjegyzØs sz vege`: ez a karakterlánc fog minden alkalommal megjelenni, amikor egy ügyfél az elérhető modulok listáját kérdezi le. Alapállapotban nincs megadva megjegyzés.
- `hosts allow = helyi.valami.org, 10.18.3.12 Øs hosts deny = *.valami.org, 10.16.3.0/241`: szükség szerint a `hosts allow` és a `hosts deny` beállításokkal hozzáférés-vezérlési listákat (ACL) adhatsz meg. Mindkét beállításhoz egymástól vesszőkkel elválasztva sorolhatod fel azokat a teljesen minősített tartományneveket vagy IP-címeket, amelyekről kifejezetten engedélyezni vagy tiltani szeretnéd a kapcsolódást. Alapértelmezett állapotban egyik beállításhoz sincs megadva semmi, vagyis minden kapcsolat engedélyezve van. Ha teljesen minősített tartománynevet adsz meg – ami a * helyettesítő karaktert is tartalmazhatja –, az rsync az IP-cím alapján visszirányú névfeloldással (reverse DNS) próbálja meg megállapítani a csatlakozó ügyfelek nevét, amit egyeztet a hozzáférés-vezérlési lista tartalmával.
  Az, hogy az rsync pontosan hogyan értelmezi ezeket a beállításokat, attól függ, hogy pontosan melyiknek adtunk értéket. Ha csak a `hosts allow` van megadva, akkor az annak értékével egyező IP-című vagy nevű ügyfelek engedélyt kapnak a csatlakozásra, a többieknek a rendszer megtiltja. Ha csak a `hosts deny` van megadva, akkor a megadottakkal egyező IP-című vagy nevű ügyfelek nem kapnak engedélyt a csatlakozásra, a többiek viszont igen. Ha mindkét beállítás szerepel, a `hosts allow` feldolgozása történik meg előbb, és ha az ügyfél IP-címe vagy neve egyező, a csatlakozást a rendszer engedélyezi.
  Ha a kérdéses IP-cím vagy név nem egyezik a `hosts allow` értékével, akkor a `hosts deny` feldolgozása következik, és ha a rendszer itt egyezést talál, akkor megtagadja a csatlakozást.
  Ha az ügyfél IP-címével vagy nevével egyik helyen sincs egyezés, a rendszer engedélyezi a csatlakozást. Jelen példában mindkét beállítás kapott értéket. Értelmezésük a következő:

- A 10.18.3.12 IP-címről érkező kéréseket a rendszer engedélyezi, de az egyéb, a 10.16.3.1-10.16.3.254 címtartományból érkezőket nem.
- A ***helyi.valami.org*** állomásról befutó kéréseket a rendszer engedélyezi, de az egyéb, a ***valami.org*** tartományból érkezőket elutasítja. Minden más kapcsolatot engedélyez.
- `ignore nonreadable = yes`: azokat a távoli fájlokat, amelyekhez az ügyfél rsync folyamatának nincs olvasási joga (lásd az `uid` és a `gid` beállításokat), a rendszer nem fogja összehasonlítani az ügyfél helyi másolatával. Ezzel a biztonság növelése helyett inkább a teljesítmény javítható, hozzáférés-vezérlési szempontból a fájlszintű jogosultságok sokkal fontosabb szerepet játszanak.
- `refuse options = checksum`: a `refuse options` beállítás arra utasítja a kiszolgálóoldali rsync folyamatot, hogy a megadott kapcsolókat hagyja figyelmen kívül, ha az ügyfél a használatukkal próbálkozna. Az rsync parancssori kapcsolói közül egyedül az ellenőrzőösszegnek van nyilvánvaló biztonsági vonatkozása. Ez arra utasítja a rendszer, hogy a normál gördülő ellenőrzőösszegek mellett az MD5-kivonatokat is számítsa ki, ami viszont nagymértékben megterheli a processzort. Letiltásával szűkül a DoS-támadási lehetőségek köre. Ugyan a tömörítést engedélyező kapcsolónak is lehet hasonló hatása, ezt a `refuse options` helyett a `dont compress` beállítással lehet szabályozni.
- `dont compress = *`: a `dont compress` beállítás révén adható meg, hogy a rendszer mely fájlokat és könyvtárakat ne tömörítse. Ha csökkenteni akarod a tömörítés használatával véghezvitt DoS-támadások esélyét, akkor a * karakterrel azt is előírhatod, hogy a rendszer semmit ne tömörítsen – így van ez az előző oldalon lévő példában is.

Az egyszerű példát áttekintve most már bátran biztosíthatsz rsync alapú fájlletöltési szolgáltatást. A következő hónapban az rsync-modulok (könyvtárak) fájlrendszerszintű, a névtelen feltöltéseket és a felhasználók azonosítását is lehetővé tévő beállításával foglalkozunk.

Linux Journal 2003. március, 107. szám

Mick Bauer (mick@visi.com)
Hálózati biztonsági tanácsadó az Upstream Solutions Inc.-nél Minneapolisban (Minnesota). Mick a szerzője a hamarosan megjelenő új O'Reilly könyvnek, amelynek címe „Building Secure With Linux”.

Java-fordítás GCJ-vel

Bár a Java nem túl népszerű az ingyenes projektek körében, a GCJ életképes választássá teszi.

A Java korántsem lett olyan átütő siker, mint a reklámhadjárat beharangozta, de azért népszerű nyelv, amelyet sok házon belüli és kiszolgálóoldali fejlesztésben, valamint más alkalmazásokban használnak. A szabad programok világában a Java már kisebb jelentőséggel bír, azonban így is számos projekt felhasználja. Ilyen Java-használó szabad projektre jó példa az Apache Foundation Jakarta projektje (➲ http://jakarta.apache.org), a W3C (➲ http://www.w3.org) néhány XML-eszköze és a Freenet (➲ http://freenet.sourceforge.net). De ez ügyben érdemes megnézni az FSF Java lapjait is (➲ http://www.gnu.org/software/java).

Az egyik oka annak, hogy viszonylag kevés Java-projekt készült, az ingyenes minőségi Java-megvalósítások vélt vagy valós hiánya. Pedig a Java születésétől fogva két megvalósítás is létezik. Az egyik a Kaffe (➲ http://www.kaffe.org), amit eredetileg ***Tim Wilkinson*** készített, és a részben általa alapított Transvirtual csapat fejleszt jelenleg is. A másik a GCJ (azaz a GNU Compiler for the Java language), amit 1996-ban indítottam útjára a Cygnus Solutionsnél. A GCC 3.0-s változatától kezdődően a GCJ beépített és támogatott GCC nyelv lett.

A hagyományos Java-megvalósításmodell

A Java-megvalósítás hagyományos módja két lépésből: a fordítási részből és a végrehajtási szakaszból tevődik össze. (E tekintetben a Java hasonlít a C-re.) A Java-programot a `javac` fordítja le, ami egy vagy több ***.class*** kiterjesztésű fájlt készít. Minden egyes ilyen fájl egyetlen osztály adatainak bináris megfelelője, ideértve az osztály eljárásainak kifejezéseit és utasításait is. Mindez bájtkóddá fordítódik, ami tulajdonképpen egy képzeletbeli veremalapú számítógép utasításkészlete. (Minthogy néhány lapka ténylegesen megérti a Java-bájtkód utasításkészletet, akár valós utasításkészletnek is nevezhetnénk.)

A végrehajtási szakaszt a Java Virtual Machine (JVM) kezeli, ami beolvassa és futtatja a ***.class***-fájlokat. A Sun változatát egyszerűen „Javának” hívják. A JVM-et egy Java-bájtkód utasításkészletű gépszimulátorának foghatjuk fel.

Az értelmező (szimulátor) elég sok végrehajtási többletköltséget okoz. A nagyteljesítményű JVM-ek esetében gyakori megoldás a dinamikus fordítás vagy a just in time (JIT) fordítók alkalmazása. Ebben az esetben a futásidejű rendszer észleli, ha egy eljárás olyan sokszor hívódik meg, hogy érdemes röptében elkészíteni az eljárás gépi kódját. Az eljárás további hívásai már közvetlenül ezt a gépi kódot futtatják.

A JIT hátránya az indulási többletköltség. Az eljárás lefordítása időbe telik, főként, ha némi egyszerűsítést is szeretnénk végrehajtani, és a fordítást mindig újra el kell végezni, valahányszor az alkalmazás elindul. Ha viszont csak azokat az eljárásokat akarjuk végrehajtani, amelyek a legtöbbször futnak, a többletköltséget ezek kikeresése fogja okozni. További gond, hogy egy jó JIT összetett és meglehetősen nagy helyet foglal el (ezenkívül az elkészített kódnak is szüksége van helyre, ami az eredeti bájtkód által használt helyen felül további helyigényt jelent). E területek kis része lehet az osztott memóriában.

A hagyományos Java-megvalósítási módszerek ezen felül nem működnek együtt túl jól más nyelvekkel. Az alkalmazásokat más módszerrel kell telepíteni (Java Archive, azaz ***.jar***-fájlokkal, és nem végrehajtható állományokkal); nagy futásidejű rendszert igényelnek, ráadásul a Java és a C/C++ közötti átjárás lassú és kényelmetlen.

A GCJ-megoldás: előrefordítás (Ahead-of-Time Compilation)

A GCJ-projekt megközelítése gyökeresen eltérő. A Javát egyszerűen egy új programozási nyelvnek tekintjük, és pontosan úgy járunk el, ahogy a többi fordításalapú nyelv esetén. Mivel a Cygnus régóta foglalkozott a GCC-vel, ami már eleve számos különféle nyelv fordítására volt képes (C, C++, Pascal, Ada, Modula2, Fortran, Chill), elég logikusnak tűnt, hogy a Javát is megpróbáljuk GCC-vel helyi kódra fordítani.

Egészében véve a Java-program lefordítása sokkal egyszerűbb, mint egy C++-program fordítása, mivel a Java nem tartalmaz se sablonokat, se előfeldolgozót. A típusrendszer, az objektummodell és a kivételkezelő modell ugyancsak egyszerűbb. A Java lefordításához a programot alapvetően egy elvont szintaktikai faként képzeljük el, ahol ugyanazokat az adatszerkezeteket használjuk, amelyeket a GCC minden egyéb nyelvhez is használ. Minden egyes Java-szerkezethez a C++ belső megfelelőjét használjuk, a többit már a GCC elvégzi.

A GCJ ezek után már élvezheti a GNU-eszközökbe épített összes egyszerűsítés és bővítmény előnyét. Ilyen egyszerűsítésre példa az általános alkifejezés-kiiktatás (sub-expression elimination), erőcsökkentés (strength reduction), ciklusegyszerűsítés (loop optimization) és a regiszterfoglalás (register allocation). Ezenkívül a GCJ kifinomultabb és időigényesebb javítást is végrehajthat, mint ami a röptében fordítók (just in time) esetében megengedhető. Akadnak olyanok is, akik vitatják ezt, mondván, a JIT jobban alkalmazkodó és testreszabottabb egyszerűsítésre is képes lehet (például a kódot a pillanatnyi végrehajtás függvényében megváltoztathatja). Meg kell hagyni, a Sun HotSpot módszere, ami ezeken az elveken alapul, igencsak lenyűgöző munkát végez. Az igazat megvallva a GCJ-vel fordított program futása valóban nem mindig lesz láthatóan gyorsabb, mintha egy JIT-alapú Java-megvalósításban futtatnánk; sőt néha akár lassabb is lehet, de ez sokkal inkább annak tudható be, hogy nem maradt időnk elkészíteni a javás egyszerűsítéseket és finomhangolni a GCJ-t. A GCJ gyakran lényegesen gyorsabb, mint egyéb JVM-ek, és egyre gyorsabb lesz, ahogy az emberek fejlesztik.

A GCJ nagy előnye az indulási sebesség és a mérsékelt memóriahasználat. Eredetileg úgy tartották, hogy a bájtkód sokkal helyhatékonyabb lesz, mint a helyi saját utasításkészlet. Bizonyos tekintetben ez valóban így is van, de nem árt, ha belegondolunk, hogy a ***.class***-fájloknak körülbelül a felét szimbolikus (nem utasítás-) adatok foglalják el. Ezek a szimbólumok minden egyes ***.class***-fájlban ismétlődnek, ugyanakkor az ELF végrehajtható állományok vagy programkönyvtárak ennél sokkal jobb közös használatot tesznek lehetővé. Ugyanakkor a bájtkód

igazán hátrányba kerül a helyi kóddal szemben a JIT-et alkalmazó JVM-ben felhasznált memória tekintetében. Ha JIT-fordítással elindítjuk a Sun JVM-jét, az alkalmazásosztályok hatalmas időmennyiséget és memóriát fognak fogyasztani, például a Sun IDE Forte for Java (ami NetBeans néven nyílt forrású változatban is elérhető) tényleg gigászi lesz. A NetBeans indítása 74 MB memóriát fogyaszt (a `top` parancs jelentése szerint), még mielőtt egyáltalán bármivel elkezdtünk volna dolgozni. A Java-alkalmazások által elfoglalt memóriaterület megnehezíti a telepítésüket is. Jól mutatja ezt az általam készített (nem túl aktív) Jemacs projekt (Jemacs.sourceforge.net), ami az Emacset Swing segítségével Java alatt próbálja meg létrehozni (no meg az alább tárgyalt Kawa felhasználásával az Emacs Lisp támogatásához). A Sun JDK1.3.1-ben indított egyszerű szerkesztőablak 26 MB-ot fogyaszt (a `top` szerint). Ezzel szemben az Xemacs mindössze 8 MB-ot eszik.

A Kawa-próbakészletet GCJ-vel és JDK1.3.1-el futtatva a GCJ nagyjából kétszer olyan gyorsnak bizonyult, nagyjából fele annyi laphibát okozott (a `time` parancs szerint), és körülbelül 25 százalékkal kevesebb memóriát fogyasztott (a `top` alapján). A próbakészlet-parancsfájl, amely a Java-környezetet többször elindítja, túlságosan sok különböző dolgot futtat, így a JIT nem tud érvényesülni (ez a JDK-ra nézve hátrányos). Ugyanakkor betölt és interaktívan futtat Scheme-kódot, így a GCJ-nek az értelmezőjén keresztül kell lefuttatnia (ami viszont a GCJ-re nézve hátrányos). Ez a kísérlet nem számíthat igazi teljesítménymérésnek, de azt azért megmutatja, hogy a GCJ-től már jelen állapotában is teljesítménynövekedést várhatunk. (Mint általában, ha a teljesítményre vagyunk kíváncsiak, futtassuk le a várható munkafajták keverékén alapuló saját teljesítménymérő tesztünket.)

A GCJ egyéb előnyökkel is rendelkezik: hibakeresés GDB-vel és csatolófelület a C/C++-hoz (később még lesz szó róla). Végül a GCJ az ipari szabványnak számító GCC-n alapuló ingyenes program, ami lehetővé teszi, hogy szabadon módosítsuk, átültessük vagy terjesszük.

Néhányan kifogásolni szokták, hogy az előfordítással (ahead-of-time compilation) elveszik a bájtkód „írjuk meg egyszer – futtassuk akárhol" hordozhatósága. Csakhogy a kötekedők figyelmen kívül hagyják azt a tényt, hogy a terjesztés nem azonos a telepítéssel. Nem azt javasoljuk, hogy a helyi kód legyen a terjesztési formátum, hacsak nem valamilyen egyedi kiépítéshez készült előregyártott csomagról van szó (pl.: RPM-ről). A Java-bájtkódot továbbra is használhatjuk terjesztési formátumként, még ha nincs is semmilyen komoly előnye a Java-forráskóddal szemben. (A Java-forráskód kevesebb együttműködési nehézséggel küzd, mint a C- vagy a C++-források.) Azt javasoljuk, hogy amikor Java-alkalmazást telepítünk, fordítsuk le a helyi kódra, ha esetleg ne lenne már eleve lefordítva.

Java-program fordítása GCJ-vel

A GCC-vel végzett Java-fordítás ismerősnek fog tűnni mindenkinek, aki már fordított C- vagy C++-programot GCC-vel. A *MyJavaProg.java* Java-program fordításához a következőket gépeljük be:

```
gcj -c -g -O MyJavaProg.java
```

Összeépítéshez használjuk az alábbi parancsot:

```
gcj --main=MyJavaProg -o MyJavaProg
➥MyJavaProg.o
```

Éppen olyan, mintha *mycxxprog.cc* C++-programot fordítanánk le:

```
g++ -c -g -O mycxxprog.cc
```

majd összeépítenénk, hogy létrehozzuk a `mycxxprog` végrehajtható állományt:

```
g++ -o mycxxprog mycxxprog.o
```

Az egyetlen új dolog a `-main=MyJavaProg` lehetőség. Erre azért van szükség, mert a Java-osztályokat gyakran `main` tagfüggvénnyel látjuk el, amit aztán kipróbálásra, esetleg önálló kis eszközként használunk. Ezért amikor egy csoport Java-fordított osztályt fordítunk egybe, több `main` tagfüggvény is lehet, az összeépítőnek (linker) meg kell mondanunk, hogy melyiket kell meghívni közülük, amikor az alkalmazás elindul. Lehetőségünk nyílik arra is, hogy Java-osztályok halmazát osztott programkönyvtárba (*.so*-fájlba) fordítsuk. Valójában a GCJ futásidejű rendszer is ilyen *.so*-fájlba fordul. Bár ennek részletei már egy másik cikk témájába tartoznának, akit érdekel, nézzen körül az alább tárgyalandó Kawa `Makefile`-jában, hogy láthassa, miként is működik.

A GCJ képességei és korlátai

A GCJ nem csupán fordítóprogram. A Sun JDK-hoz hasonló teljes körű Java-környezetnek szánták. Ha megadjuk a `-C` kapcsolót, a `gcj` hagyományos *.class*-fájlokat fordít. A cél az volt, hogy a `gcj -C` parancsot a sunos `javac` parancs kiváltására is fel lehessen használni.

A GCJ-ben megtalálhatjuk a bájtkódértelmezőt (***Kresten Krab Thorup*** jóvoltából) és a teljes értékű `ClassLoadert`. Az önálló `gij` program a Sun Java-programjának kiváltására használható. A GCJ a GCC 3.0-ban megjelent `libgcj`-t használja. A futásidejű könyvtárban megtalálhatjuk a futásidejű támogatás magvát, ***Hans Boehm*** figyelemre méltó hagyományos szemétgyűjtőjét, a bájtkódértelmezőt, illetve az osztályok jókora könyvtárát. Jogi és technikai szempontok miatt a GCJ nem tartalmazhatja a Sun osztálykönyvtárát, ezért saját változattal bír. A GNU Classpath Projekt ugyanazt az engedélyt és FSF jogvédelmét alkalmazza, mint amit a `libgcj` és `libstdc++` használ, így a két projekt osztályai egyesíthetők. Mi a GPL-t használjuk, azzal a különleges kitétellel, hogyha végrehajtható állományt a `libgcj`-t más fájlokkal egybefordítva készítünk, az önmagában még nem feltétlenül jelenti azt, hogy az eredménynek is GPL alá kell tartoznia. Így akár üzleti programok is készíthetők a szabványos C++ vagy Java futásidejű könyvtárakkal.

A `libgcj` könyvtár a GUI nélküli alkalmazásokhoz szükséges legtöbb szabványos osztályt tartalmazza, ide értve a ***java.lang***, ***java.io***, ***java.util***, ***java.net***, ***java.security***, ***java.sql*** és a ***java.math*** osztályok csomagjait. A legfontosabb hiányzó osztályok a grafika készítését támogató AWT- vagy Swing-bővítmények. A legtöbb magas szintű AWT osztály készen van, de az alacsony szintű osztályok még nincsenek olyan fokon, hogy érdemben használni lehessen őket. Várjuk a segítő szándékú jelentkezőket!

Együttműködés a C/C++-rendszerekkel

Bár Javában sok mindent meg tudunk csinálni, néha mégis szükség lehet rá, hogy más nyelvekben írt programkönyvtárakat hívjunk meg. Ennek az egyik oka az lehet, ha olyan alacsony szintű kódot akarunk használni, amit a Javában nem lehet elkészíteni, de ki akarjuk használni egy olyan könyvtár lehetőségeit, amit nem szeretnénk újraírni, vagy sebességre menő alacsonyszintű kódolást végzünk. Mindezt megtehetjük egyszerűen úgy, hogy őshonosnak (native) jelölünk meg néhány Java-tagfüggvényt, majd a tagfüggvény testének létrehozása helyett a megvalósítást

valamilyen más nyelven készítjük el. A Sun 1997-ben kiadta a Java Native Interface (JNI) csatolófelületét, ami a C vagy C++ nyelveken készíthető őshonos tagfüggvények készítését szabályozó szabvány. A JNI legfőbb célja az átültethetőség, azazhogy az egyik Java-megvalósításhoz készült őshonos tagfüggvény a másik Java-megvalósításon újrafordítás nélkül futtatható legyen. Ez azonban a zárt forrású, terjesztett bináris világ céljaihoz készült, és jóval kevesebb haszna van a szabad programok világában, legfőképpen azért, mert újra kell fordítanunk, ha processzort vagy operációs rendszert váltunk.
A megfelelőség (compatibility) érdekében JNI alatt mindent közvetett módon, függvénytáblákon keresztül végzünk el. Ez azonban igencsak lelassítja a JNI-t. Még ennél is kellemetlenebb, hogy az összes ilyen függvényt fárasztó és hibákra hajlamos szabályok alapján kell létrehoznunk. Bár a GCJ támogatja a JNI-t, választási lehetőséget is nyújt. A Compiled Native Interface (CNI, amelyet egyben Cygnus Native Interface-ként is olvashatunk) azon az elképzelésen alapul, miszerint a Java alapvetően a C++ alhalmaza, és a GCC ugyanazt a hívási módszert alkalmazza mind a C++, mind a Java esetében. Így természetesen lehetséges C++ nyelvet használó Java-tagfüggvényt írni, és a C++ szabályos írásmódját felhasználva érhetjük el a Java-mezőket, hívhatjuk meg a Java tagfüggvényeket. Mivel ugyanazt a hívási elgondolást és adatkiosztást alkalmazzák, a C++ és a Java között nincsen szükség átalakításra vagy mágikus bűvészkedésre.
A CNI- és JNI-példák bemutatása már egy következő cikk témája lesz. A GCJ kézikönyv elég jól ismerteti a CNI-t, (➲ http://gcc.gnu.org/onlinedocs/gcj) a libgcj forrásokban pedig számos példát találhatunk.

Kawa: sémák fordítása Javára

A Java-bájtkód a Java-programok eléggé közvetlen kódolása, amit nemigen terveztek bármi másra. Ennek ellenére használják más nyelven írt programok kódolására is.
A ➲ http://grunge.cs.tu-berlin.de/~tolk/vmlanguages.html címen olvasható listában megnézhetjük hogy, milyen más programozási nyelveket készítettek a Java fölé. A legtöbb ezek közül értelmező, de néhány közülük ténylegesen bájtkódot készít. Az első egy az egyben felhasználhatja a GCJ-t; az utóbbi esetleg a GCJ-t használhatja a natív kód fordítására.
Az egyik ilyen fordító a Kawa, amit 1996 óta fejlesztek. A Kawa a Javát használó nyelvek megvalósítására használható eszközkészlet és egyúttal a Scheme programozási nyelv megvalósítása. A GCJ segítségével minden fizetős program nélkül elkészíthetjük a Kawát. A Kawa honlapon (➲ http://www.gnu.org/software/kawa) megtaláljuk a GCJ és Kawa letöltésére és telepítésére vonatkozó utasításokat.
A Kawa felhasználói beavatkozást igénylő (interactive) módban is használható. Most megadunk egy faktoriális függvényt, majd meghívjuk:

```
$ kawa
#|kawa:1|# (define (factorial x)
#|(---:2|#   (if (< x 2) x (* x (factorial
➥(- x 1)))))
#|kawa:3|# (factorial 30)
265252859812191058636308480000000
```

A dolognak az az érdekessége, hogy a faktoriális függvényt a Kawa bájtkóddá fordította, és mint új osztályt azonnal be is töltötte. A folyamat a Java `ClassLoader` szerkezetét használja az új osztály futásidejű meghatározásához, létrehozva az osztály bájtkódját tartalmazó bájttömböt. Az új osztály eljárásai a GCJ bájtkódértelmezőjén keresztül futnak le.
Természetesen a kódot általában kényelmesebb fájlba helyezni:

```
$ cat > factorial.scm
(define (factorial x)
(if (< x 2) x (* x (factorial (- x 1)))))
(format #t "Factorial ~d is ~d.~%~!" 30
➥(factorial 30))
^D
$ kawa -f factorial.scm
Factorial 30 is
265252859812191058636308480000000.
```

Megnövelhetjük a Scheme-kód teljesítményét, ha a Kawa segítségével előrefordítjuk, egy vagy több ***.class***-fájlt hozva létre:

```
$ kawa --main -C factorial.scm
(compiling factorial.scm)
```

A lefordított fájlt aztán betölthetjük:

```
$ kawa -f factorial.class
Factorial 30 is
➥265252859812191058636308480000000.
```

Az osztályfájl helyi kódra történő fordításához a `gckawa`-t használhatjuk, ami nem más, mint a szükséges környezeti változókat (`LD_LIBRARY_PATH` és `CLASSPATH`) beállító, majd a GCJ-t meghívó parancsfájl:

```
$ gckawa -o factorial --main=factorial
➥-g -O factorial*.class
```

A ***factorial*.class*** névben használt helyettesítő (joker) karakter használata itt szükségtelen, de hasznos lehet, ha a Kawa több ***.class***-fájlt készít.
Ezt követően végrehajthatjuk az eredményül kapott normális GNU/Linux ELF végrehajtható állományt. A `libgcj.so` (GCJ futásidejű könyvtár) és a `libkawa.so` (a Kawa futásidejű könyvtár) kapcsolódik hozzá.
Más nyelvekhez is használható ugyanez a megközelítés. Jelenleg a W3C új XML-lekérdező nyelvének, az Xquery-nek megvalósításán dolgozom a Kawa segítségével. A GCJ-vel készült alkalmazások közé tartozik néhány Apache-modul, a GNU-Paperclips és a Jigsaw.

Összegzés

A GCJ fejlesztése mostanában igen nagy pezsgést mutat, és számos feladathoz megbízható alapot jelenthet. Reméljük, hogy sikerült kedvet csinálni a Javának a szabad programjainkban történő felhasználásához, a GCJ-hez mint előnyben részesített Java-megvalósításhoz, illetve néhányan talán segítenek majd még jobbá tenni a GCJ-t.

Linux Journal 2003. január, 105. szám

Per Bothner
Az 1980-as évektől kezdve GNU és programon dolgozik. A Cygnusnál a GCJ projekt műszaki vezetője volt. Jelenleg független szakértő.
A ➲ http://www.bothner.com/per címen érhető el.

Az Intel fordító működése

Hogyan győzte le méréseink során az Intel fordítója a gcc-t?

A Linux egyre növekvő szerepének következtében a fejlesztők és a kutatók egyre több fejlesztőeszközt igényelnek. Ebből az okból született meg az Intel legújabb, kiválóan testreszabható C++- és Fortran-fordítója is, amit kifejezetten olyan alkalmazásfejlesztőknek szánnak, akik az Intel IA-32 és Itanium rendszerein dolgoznak. Ezek a fordítók támogatják a szigorú ANSI szabványt és néhány más népszerű kiterjesztést is. Ez a cikk az Intel IA-32-es fordítójának a tulajdonságaival és hatékonyabbá tételével foglalkozik. A cikk hátralévő részében az Intel IA-32-es C++- és Fortran-fordítóira csak „Intel-fordító”-ként fogunk hivatkozni.
Az Intel fordítója a program minden szintjén megpróbálja feljavítani annak képességeit: a magas szintű ciklusoktól és függvényhívásoktól kezdve egészen a fordító adatfolyamának hatékonyabbá tételéig és az alacsonyszintű változtatásokig – ilyen például az utasításütemezés, az alapszintű szerkezetkialakítás és a regiszterfoglalás. Ebben a cikkben elsősorban csak az Intel fordítójára jellemző hatékonysági jellemzőkkel foglalkozunk.

Hagyományos hatékonyságnövelő ötletek

A két leggyakrabban használt hatékonyságnövelő módszer a sokszor végrehajtódó utasítások számának csökkentése és az utasítások gyorsabb változataikra történő cserélése. Ebbe a két csoportba számos hagyományos egyszerűsítő eljárás tartozik: a másolás és változó híresztelés, a felesleges kifejezések eltüntetése, a felhasználatlan kódok eltávolítása, az átlátások megszüntetése, a függvénybeszúrás, a záró műveletek többszöri ismétlődésének eltakarítása és így tovább.
Az Intel fordítója mindkét esetre rengeteg egyszerűsítő módszert tud. Rengeteg helyi egyszerűsítés az úgynevezett SSA-n, vagyis a maradandó egyszeri hozzárendelésen alapul. A felesleges (vagy részlegesen felesleges) kifejezések eltávolításáért felel többek között a Chow-algoritmus is (lásd az ***Irodalomjegyzék*** 6. pontjában), ami akkor tekint egy kifejezést feleslegesnek, ha szükségtelenül többször kerül kiértékelésre egy végrehajtási szakasz során. Például a következő állításban:

```
x[i] += a[i+j*n] + b[i+j*n];
```

Az `i+j*n` kifejezés kétszeri kiértékelése felesleges. A részleges ismétlődés azt jelenti, hogy egy kifejezés valamelyik része bizonyos végrehajtási szakaszokon többször ismétlődik, de nem feltétlenül az összes végrehajtási szakaszon. Vegyük a következő példát:

```
if (c) {
    x = y+a*b;
} else {
    x = a;
}
z = a*b;
```

Az `a*b` kifejezés részlegesen felesleges. Ha az `else` hajtódik végre, az `a*b` kifejezés csak egyszer értékelődik ki; ha azonban az `if` ág hajtódik végre, akkor a kifejezés kétszer számolódik ki. A kódot így módosíthatjuk:

```
t = a*b;
if (c) {
    x = y+t;
} else {
    x = a;
}
z = t;
```

Így az `a*b` eredménye csak egyszer értékelődik ki, függetlenül attól, hogy melyik ág hajtódott végre.
Ezt az átalakítást megfontoltan kell használnunk, mivel az ideiglenes változók számának növekedése, amelyek elvileg regiszterekben tárolódnak, az élettartam növekedéséhez vezethetnek, ezáltal a regiszterek számának a csökkenése következik be. Egy másik algoritmussal (az ***Irodalomjegyzék*** 9. pontja) – ami nagyon hasonlít Chow algoritmusához – a nem használt tárolókat szüntethetjük meg, amiket még a lekérdezésük előtt egy másik tároló vált fel, illetve a részlegesen használt tárolókat, amik csak bizonyos futási szakaszokon nem kerülnek felhasználásra. Az SSA-n alapuló egyéb hatékonyságnövelő eljárások a változó- és feltételkiértékelésekre vonatkoznak. Vegyük a következő példát:

```
if (x>0) {
    if (y>0) {
        . . .
        if (x == 0) {
            . . .
        }
    }
}
```

A legkülső feltételből egyértelműen kiderül, hogy `x` nem lehet egyenlő nullával, így a belső feltétel értelmetlen, ezért kiértékelése szükségtelen. Az előző példák talán mesterkéltnek tűnhetnek, de gyakran adódhatnak hasonló helyzetek címek kiszámításakor, makrókban vagy beszúrt függvényekben.
Az Intel fordítója erőteljes memória-összezavarást alkalmaz, hogy észrevegye, ha bizonyos memóriahivatkozások esetleg átfedik egymást. Ez az elemzés fokozza például a regiszterkihasználást, és lehetővé teszi a kódban lévő párhuzamosságok észlelését és kiaknázását – erre a következőkben még bővebben kitérünk. Az Intel fordítója az eljárásközi hívásokat is egyszerűbbé teszi, ebbe beletartozik az önműködő és kézi függvénybeszúrás vagy a részleges függvénybeszúrás, ahol a függvényből csak a szükséges részek kerülnek be. A fordító egyszerűbbé teszi az eljárásközi állandókat és a kivételkezelést is. Ha a „teljes program” elemzést bekapcsoljuk, bizonyos adatszerkezetek elhelyezkedése megváltozhat – például a Fortran COMMON BLOCK-jai –, hogy a memóriaelérés az adott processzoron hatékonyabbá váljon. Például az adatok elhelyezését

különféle határokhoz lehet igazítani. Ráadásul az Intel fordítója még okosabb döntéseket hozhat, meg tudja határozni, hogy mikor és hol van szükség egy-egy beszúrásra. Ehhez kétféle kialakítási eljárást használ fel: a nyugvó kialakítást és a változó kialakítást. A nyugvó kialakítás olyan adatokra hivatkozik, amik fordítási időben megbecsülhetők vagy kikövetkeztethetők, míg a változó kialakítás a program futása közben szerzett adatoknak megfelelően alakul. A kialakítás e két fajtájáról lesz szó a következő részben.

A kialakítás egyszerűsítése

Elsőként vegyük a nyugvó kialakítást, amire ebben a kódrészletben láthatunk példát:

```
g();
for (i=0; i<10; i++) {
    g();
}
```

Nyilvánvalóan a cikluson belüli hívás tízszer többször hajtódik végre, mint a cikluson kívüli hívás. Ezt sok esetben azonban nem lehet ilyen jól megbecsülni, mint ahogyan a következő példában sem:

```
for (i=0; i<10; i++) {
    if (feltØtel) {
        g();
    } else {
        h();
    }
}
```

Itt nehéz megmondani, hogy melyik függvény fog nagy valószínűséggel végrehajtódni. Ha a `h()` eljárás a program futásának megszakítását eredményezné vagy valamilyen egyéb olyan eljárás volna, ami nem tér vissza, akkor az első ág, vagyis a `g()` eljárás beszúrásával többre mennénk, mivel az valószínűleg többször hajtódna végre. Ilyen értesülések hiányában azonban rendkívül nehéz eldönteni, hogy melyik eljárást szúrjuk be, esetleg mindkettővel megtegyük-e. A másik lehetőség, hogy változó kialakítást alkalmazunk.

A változó kialakítás a program végrehajtása során nyeri az adatait. Ezzel a fordítót olyan előnyhöz juttatjuk, hogy módjában áll megismerni a program futásának menetét, és ennek megfelelően tudja egyszerűsíteni. Egy háromlépéses művelet részeként az alkalmazást először a kialakítást feltérképező műveletekkel együtt fordítjuk. Ezt követően a programot mintaadatokkal futtatjuk le, amiből a fordító számára összeáll egy olyan adatbázis, amit egy újabb fordítás alkalmával felhasználhat. Végül az adatbázisban található adatok segítségével a fordító már könnyen meghatározhatja, hogy melyik gyakran végrehajtódó programrészt helyezze egy csoportba, vagy a kód melyik részét kell átrendeznie, milyen függvényeket kell teljesen vagy részlegesen beszúrnia, illetve hogyan alakítsa ki a regiszterek lefoglalását. A regiszterfoglalás az Intel fordítójában gráfalapú egyesítésekre épül (lásd még az ***Irodalomjegyzék*** 5. pontjában), ami a kódot régiókba tagolja. Ezek a régiók általában ciklusok törzsei vagy egyéb összetartozó egységek. A kialakítási adatok birtokában a régiók hatékonyabban kiválaszthatók, és a programrészek tényleges végrehajtódásának a számán alapulnak, nem találgatásokon. Ilyen módon a programtörmelékek a program ritkábban futtatott területeire helyezhetők.

Regiszteren belüli csoportosítás

A jelenlegi gépeken a párhuzamosan futtatható műveletek feltárása az egyszerűsítés érdekében rendkívül fontos. Az Intel fordítója kulcsszerepet játszhat a programon belüli párhuzamosan futtatható műveletek megtalálásáért folytatott erőfeszítésekben, úgy, hogy megkönnyíti az olyan hatékonyságnövelő eljárásokat, mint az önműködő összegyűjtés, az önműködő párhuzamosítás, vagy az OpenMP parancsainak a támogatása. Nézzünk meg egy példát, amiben azt mutatjuk be, miként alakul át a soros ciklus önmagától egy olyan formátumba, ami az Intel MMX és SSE/SSE2 (folyamatos SIMD-kiterjesztés) megoldásait képes kihasználni. Ezt a folyamatot „regiszteren belüli csoportosításnak" (Intra-Register Vectorization) hívjuk (lásd az ***Irodalomjegyzék*** 1. pontjában). Példánkban a következő függvény adott:

```
void vecadd(float a[], float b[],
➥float c[], int n)
{
  int i;
  for (i = 0; i < n; i++) {
       c[i] = a[i] + b[i];
  }
}
```

Az Intel-fordító az `addps` utasítás segítségével a ciklust négy egyszeres pontosságú lebegőpontos műveletté alakítja át, amelyek egy időben hajtódnak végre. Ilyenformán az eredmény a következőképpen nézne ki:

```
for (i = 0; i < n; i+=4) {
     c[i:i+3] = a[i:i+3] + b[i:i+3];
}
```

A skaláris eltakarító ciklus végrehajtaná a fennmaradó utasításokat, ha az `n` lépésszámláló nem lenne pontosan osztható néggyel. Ebben a lépésben több folyamat vesz részt. Elsőként – mivel elképzelhető, hogy a tömbök kezdőcímeiről nem áll rendelkezésre adat – egy futásidejű kódot kell beszúrni, hogy meg lehessen győződni arról, hogy a tömbök adatterületei nem fedik-e egymást (önműködő függéspróba), és hogy a ciklus fennmaradó része 16 bájtos határokra igazított címekkel fut-e (önműködő cikluscsupaszítás a megfelelő elrendezéshez).

A hatékony csoportosítás érdekében csakis ciklusokat vagy megfelelő hosszúságú kódrészeket csoportosítunk. Ha az ismétlésszám túl kicsi, akkor egy egyszerű soros ciklust használunk. Az egyszerű ciklusok mellett a csoportosító támogatja a csökkenő elemű ciklusokat is (mint amilyen például a számokból álló tömb összegzése, vagy egy tömb legkisebb és legnagyobb elemének megkeresése, feltételes szerkezetek, szórási számtan és egyéb kifejezések). Még a trigonometrikus matematikai függvények csoportosítása is lehetséges a csoportosítási matematikai könyvtár segítségével.

Hogy érzékelni lehessen a regiszteren belüli csoportosításnak köszönhető teljesítménynövekedést, a Linpack nevű mérőprogrammal (ez a ➲ http://www.netlib.org/benchmark címről tölthető le, C és Fortran nyelvű) kétszeres pontossággal lemértük a teljesítménykülönbséget. Ez a mérés egy lineárisegyenlet-megoldó teljesítményét méri, ami a tényezőkre bontáshoz és a megoldáshoz a `DGEFA` és a `DGESL` eljárások egyikét használja fel. A mérés főként abból áll, hogy – amíg a tényezők keresése folyik – az egyes együtthatók mátrixának oszlopain ismételten hívogatjuk az elsőszintű `BLAS DAXPY` eljárást. Alapbeállítások-

kal (`-O2`-es kapcsoló), egy 2,66 GHz Pentium 4-es processzorral a mérés eredménye 1,049 MFLOPS egy 100×100-as táblázat kiszámításakor. Ha engedélyezzük a regiszteren belüli csoportosítást (`-xW` kapcsoló) a Pentium 4-es processzorhoz, a teljesítmény egészen 1,292 MFLOPS-ig nő, ami összesen húszszázaléknyi teljesítménynövekedést jelent.

Az OpenMP és az önműködő párhuzamosítás

Az OpenMP a közelmúltban a C/C++ és Fortran nyelvű osztottmemória-kezelésen alapuló párhuzamos programozás szabványává vált. A felhasználó számára anélkül lehetővé teszi a párhuzamos futtatást, hogy a felhasználónak el kellene merülnie a művelet részleteiben: az ismétlések felosztásában, az adatmegosztásban, a szálak időzítésében vagy az összehangolásban. Ezeket a beállításokat alapul véve az Intel fordítója képes önműködően bármilyen kódból párhuzamosan futó kódot előállítani. Az Intel-fordító támogatja az OpenMP C++ 2.0 és OpenMP Fortran 2.0 szabványos beállításokat a párhuzamosság tulajdonságainak meghatározására. Ennek segítségével a többprocesszoros rendszereken az alkalmazások a lehető legnagyobb teljesítmény elérése érdekében igényeiknek megfelelően és pontosan állíthatják be a párhuzamossági tulajdonságokat.
A következőkben egy OpenMP-s beállításokat felhasználó példaprogramot láthatunk az Intel C++ Linux OpenMP fordítóhoz:

```
#define N 10000
void   ploop(void)
{
  int k, x[N], y[N], z[N];
  #gyakorlati pØlda az omp pÆrhuzamos tÆsra
  ➥zÆrt(k) osztott(x,y,z)
  for (k=0;  k<N; k++) {
    x[k] = x[k] * y[k] + workunit(z[k]);
  }
}
```

A `for` ciklust a szálak egy csoportja egymás között felosztva fogja végrehajtani. A `k` változó `private` (zárt), vagyis minden szál saját példányt kap belőle, míg az `x`, `y`, `z` változókat a szálak közösen használják.
A létrejövő többszálú programkód alább látható. Az Intelfordító futásidejű OpenMP hívásokat hoz létre, amik a szálak létrehozásáért, kezeléséért és összehangolásáért (lásd az ***Irodalomjegyzék*** 1–2. pontjában) felelősek.

```
#define N 10000
void  ploop(void)
{
    int k, x[N], y[N], z[N];
    __kmpc_fork_call(loc, 3,
    ➥T-entry(_ploop_par_loop), x, y, z)
    goto L1:
    T-entry _ploop_par_loop(loc, tid, x[],
    ➥y[], z[]) {
       lower_k = 0;
       upper_k = N;
       __kmpc_for_static_init(loc, tid,
       ➥STATIC, &lower_k, &upper_k, ...);
       for (local_k=lower_k;
       ➥local_k<=upper_k;
            local_k++)  {
          x[local_k] = x[local_k] *
          ➥y[local_k]+ workunit(z[local_k]);
       }
       __kmpc_for_static_fini(loc, tid);
       T-return;
    }
L1: return;
}
```

A többszálú kódkezelő beszúrja a szállétrehozó `__kpmc_fork_call` függvényt – `T-entry` belépési ponttal és adatkörnyezettel (például `tid`, thread id, vagyis szálazonosító) minden egyes ciklushoz. Ezzel belép az Intel OpenMP könyvtárába, és létrehoz néhány szálat a `for` ciklus párhuzamos futtatására.
Az OpenMP-beállításokkal kísért soros ciklus többszálú kóddá alakul úgy, hogy a ciklus alsó és felső határolói helyivé alakulnak, illetve a ciklus léptető változója is egyedivé válik. Végül létrejön egy többszálú futásidejű kezdeményező és összehangoló kód minden egyes `[T-entry, T-ret]` pár által meghatározott `T-region`-hoz. A `__kpmc_for_static_init` hívás kiszámolja a helyi ciklus alsó és felső határát, és az időzítő beállításoknak megfelelően sorra lépked az egyes szálakon. A példakódunkban állandó időzítést használunk.
A `__kmpc_for_static_fini` lib-hívás futásidőben értesíti a rendszert, hogy az éppen futó szál végzett egy ciklussal. Ahelyett, hogy a fordító teljes forrás–forrás-átalakítást végezne, mint ahogyan azt más fordítók teszik, például az OpenMP NanosCompiler és az OdinMP, az Intel fordítója ezeket a változásokat beépítetten kezeli. Ez egyprocesszoros rendszerek esetében lehetővé teszi az együttműködést az OpenMP-n kívül más fejlett, magas szintű fordítói módszerekkel, például a csoportosítás és ciklusátalakítások során.
Mindezek mellett a fordító lehetővé teszi az OpenMP-irányítókkal egyénileg beállított párhuzamosságot is, illetve a `-parallel` kapcsolóval a fordító önműködően hoz létre párhuzamos kódot. Ennek a kapcsolónak a megadásával a fordító önmaga ellenőrzi, hogy milyen ciklusok párhuzamosíthatók, vagyis milyen ciklusok nem tartalmaznak ciklusszintű függőségeket. Az önműködő párhuzamosítás a fejlett memóriatisztázó módszert használja fel elemzéséhez, illetve a saját szerkezetelemzőjét, hogy eldöntse, mikor szükséges a párhuzamosítás.

Processzorazonosítás

Az Intel-fordító processzorazonosító tulajdonsággal bír, aminek a segítségével egy-egy elem többféle IA-32 rendszeren is felhasználható – vagy a kézi processzorazonosító, vagy az önműködő processzorazonosító segítségével. A kézi processzorazonosító használatával egy függvénynek többféle változatát is megírhatjuk, amelyek mindegyike kifejezetten egyfajta IA-32-es rendszerhez kötődik, vagy pedig általános érvényű, tehát minden IA-32-es rendszeren használható. Az Intel-fordító létrehoz egy kódot, ami önműködően eldönti, hogy az alkalmazás milyenfajta processzoron fut, és ennek megfelelően választja ki egy-egy függvénynek azt a változatát, amit felhasznál. Ezzel a futásidejű meghatározó rendszerrel a programozó számára lehetővé válik, hogy az alkalmazásába bizonyos rendszerfüggő lehetőségeket építsen be, mint amilyen például az SSE vagy az SSE2, anélkül, hogy feláldozná a rugalmasságot. Továbbá lehetőséget ad egyazon program különböző rendszereken történő futtatására, mégha azok nem is támogatnak bizonyos utasításokat.
Az önműködő processzormeghatározó is hasonlóan működik, azzal a különbséggel, hogy ebben az esetben a fordító az egyes függvények különböző változatait maga hozza létre. A fordítás

alatt a fordító eldönti, hogy különböző rendszerfüggő beállításokkal milyen függvényekből lehet jobb teljesítményt kihozni. Ezt követően ezek a függvények önműködően több változatban jönnek létre. Egy változat készül, ami tartalmazza a rendszerfüggő utasításokat, egy pedig, ami csak az általános utasításokat tartalmazza. Ennek a szolgáltatásnak az az előnye, hogy a programozónak nem kell újraírnia a kódot. Egy egyszerű forrásfájl esetében egy parancssori kapcsolóval megadható az önműködő processzorazonosítás bekapcsolása. Vegyük például a következő függvényt:

```
void init(float b[], double c[], int n)
{
  int i;
  for (i = 0; i < n; i++) {
       b[i] = (float)i;
  }
  for (i = 0; i < n; i++) {
       c[i] = (double)i;
  }
}
```

Az Intel fordítója ennek a függvénynek akár három változatát is létrehozhatja. Készül egy általános változat, ami minden IA-32-es processzoron képes futni. Készül egy változat, ami a Pentium III-as processzor SSE-re épülő lehetőségeit használja ki, és készülhet egy harmadik változat, ami a Pentium 4-es processzort aknázza ki úgy, hogy mindkét ciklust csoportosítja, és felhasználja az SSE2-es utasításokat. A létrejövő függvény a következő azonosítókóddal kezdődik:

```
.L1  testl     $-512, __intel_cpu_indicator
     jne       init.J
     testl     $-128, __intel_cpu_indicator
     jne       init.H
     testl     $-1, __intel_cpu_indicator
     jne       init.A
     call      __intel_cpu_indicator_init
     jmp       .L1
```

A kódban az `init.A`, `init.H` és `init.J` függvény az általános, az SSE és SSE2-es változat, ebben a sorrendben.

Nyelvi kiterjesztések

Amellett, hogy az Intel-fordító szigorúan megfelel az ANSI követelményeinek, léteznek kapcsolók, amelyek számos GCC-lehetőséget fednek le, például a `long long int`, a 0 hosszúságú tömbök és a változó mennyiségű kapcsolóval rendelkező makróhívások. A GCC stílusú beszúrt assembly kód úgyszintén támogatott. A fordító támogatja a DWARF2-es hibakövetési információkat, így a létrejövő programok a GDB-vel is nyomon követhetők. Bizonyos microsoftos kiterjesztések is engedélyezettek, mint például a `__declspec` tulajdonságok, és a Microsoft stílusú beszúrt assembly-kód.
A beszúrt assembly mellett az Intel-fordító beépítetten támogatja az MMX és az SSE/SSE2-es utasításkészleteket is. Így a fordító hozzáfér a rendszerfüggő lehetőségekhez, anélkül, hogy ez bármilyen teljesítménybeli visszaeséssel vagy hibás alkalmazással járna, ami gyakran abból adódik, hogy a beszúrt assembly-részek összeütközésbe kerülnek az Intel fordítója által alkalmazott elemző és átalakító utasításokkal. A beépített lehetőségek felhasználásával a programozó kiaknázhatja a processzorban rejlő lehetőségeket, ugyanakkor élvezheti a regiszterfoglalás, az ütemezés és az egyéb szolgáltatások előnyeit is.

Köszönetnyilvánítás

A szerzők külön köszönetet mondanak ***Zia Ansari***-nak és ***David Kreitzer***-nek azért a segítségért, amit a fordító részleteinek a megértéséhez nyújtottak. Ezenkívül köszönetünket fejezzük ki az Intel-fordítót létrehozó csapat minden tagjának.

Linux Journal 2003. február, 106. szám

Dale Schouten (Dale.A.Schouten@intel.com)
Az Intel-fordítólaborban dolgozik. Az Illinois-i Egyetemen szerzett PhD fokozatot. Dale egyúttal nem hivatásos zenész, és édesapja két különleges gyereknek.

Xinmin Tian (Xinmin.Tian@intel.com)
Szintén az Intel-fordítólaborban dolgozik. Ő irányítja az OpenMP párhuzamosítási csoportot. A Tsinghua Egyetemen számítógépes tudományok terén sorrendben BSc, MSc és PhD fokozatot szerzett.

Aart Bik (Aart.Bik@intel.com)
A számítógépes tudományok terén elért MSc fokozatát az Utrechti Egyetemen kapta, míg PhD fokozatát a Leideni Egyetemen érte el. Jelenleg az Intel-fordítólaborban párhuzamosításon és csoportosításon dolgozik.

Milind Girkar (Milind.Girkar@intel.com)
A számítógépes tudományok terén elért PhD fokozatát az Illinoisi Egyetemen kapta. Jelenleg az IA-32 fordító fejlesztési csoportot irányítja az Intel-fordítólaborban.

Irodalomjegyzék

1. *Aart Bik*, *Milind Girkar*, *Paul Grey* és *Xinmin Tian*: „Efficient Exploitation of Parallelism on Pentium III and Pentium 4 Processor-Based Systems”, Intel Technology Journal, Q1, 2001.
2. *Xinmin Tian*, *Aart Bik*, *Milind Girkar*, *Paul Grey*, *Hideki Saito* és *Ernesto Su*, „Intel OpenMP C++/Fortran Compiler for Hyper-Threading Technology: Implementation and Performance”, Intel Technology Journal, Vol. 6, Q1, 2002.
3. *E. Ayguade*: „NanosCompiler: A Research Platform for OpenMP Extensions”, In Proc. of the First European Workshop on OpenMP, 1999. október
4. *Brunschen* és *M. Brorsson*: „OdinMP/CCp – A Portable Implementation of OpenMP for C”, In Proc of the 1st European Workshop on OpenMP (EWOMP '99), 1999. szeptember
5. Guei-Yuan Lueh, Thomas Gross és Ali-Reza Adl-Tabatabai: „Global Register Allocation Based on Graph Fusion” LCPC, 1996
6. F. Chow, et al. New Algorithm for Partial Redundancy Elimination based on SSA Form. PLDI, 1997
7. *M. Wegman* és *K. Zadeck*: „Constant Propagation with Conditional Branches” ACM TOPLAS, 1991. április
8. *Rakesh Ghiya*, *Daniel Lavery* és *David Sehr*: „On the Importance of Points-To Analysis and Other Memory Disambiguation Methods for C Programs” PLDI, 2001
9. *R. Lo, F. Chow, R. Kennedy, S. Liu and P. Tu:* „Register Promotion by Sparse Partial Redundancy Elimination of Loads and Stores. Proceedings of the ACM SIGPLAN” Conference on Programming Language Design and Implementation, pp. 26–37., 1998. június
10. Intel C- és C++-fordítók Linuxhoz: ➲ http://developer.intel.com/software/products

Az USB-soros illesztőprogram-réteg (2. rész)

Hogyan készíthetsz olyan USB-s eszközt, ami az általános USB-soros illesztőprogrammal használható? Olvass tovább, és hamarosan megtudod!

Első részében írásunknak (Linuxvilág, 2003. március) bemutattam az USB-soros réteget, illetve azokat az alapvető műveleteket, amelyekkel egy illesztőprogram a rétegnél bejegyezhető. Ez alkalommal arról lesz szó, hogy az adatok miként mozognak a rétegen keresztül, illetve hogyan jelennek meg az USB-soros eszközök `sysfs` alatt.

Általános USB-soros eszközök

Az első részben említettem, hogy az általános USB-illesztőprogram segítségével könnyedén, egyedi rendszermagszintű programozás nélkül is munkára foghatunk egy USB-s eszközt. Sajnos arról elfelejtettem szót ejteni, hogy pontosan milyen módon is kell ezt véghezvinni – szerencsére az olvasóktól érkezett kérdések hamar egyértelművé tették: nem sumákolhatok.
Ha az általános USB-soros illesztőprogrammal használható USB-s eszközt szeretnél létrehozni, mindössze két tömeges adatmozgatásra alkalmas USB-végpontra van szükséged, egy ki- és egy bemenetre. Az általános USB-soros illesztőprogram ezt a két végpontot egyetlen tty-eszközzé fogja össze, amiről olvasni, illetve amire írni lehet a felhasználói területről. Például egy eszköz végpontjainak a leírói a */proc/bus/usb/devices* alatt találhatók, ez maga mégis egykapus eszközként kezelhető. Csatlakoztatáskor az alábbi rendszermagüzeneteket láthatjuk:

```
Generic converter detected
(A rendszer ÆltalÆnos Ætalak t t ismert fel.)
Generic converter now attached to ttyUSB0
(Az ÆltalÆnos Ætalak t  a ttyUSB0 csom ponton
csatlakozik.) (devfs hasznÆlatakor usb/tts/0)
```

Ezt követően a */dev/ttyUSB0* csomóponton keresztül bármely felhasználó adatokat küldhet az eszköznek.
Ha egy eszköz egynél több tömeges adatmozgatásra szolgáló ki- és bemenetpárral rendelkezik, akkor a rendszer több kaput is rendel hozzá. Például: egy eszköz végpontjainak leírói a */proc/bus/usb/devices* alatt találhatók, az eszköz pedig kétkapusként jelenik meg. Csatlakoztatáskor az alábbi rendszermagüzenetek láthatók:

```
Generic converter detected
(A rendszer ÆltalÆnos Ætalak t t ismert fel.)
Generic converter now attached to ttyUSB0
(Az ÆltalÆnos Ætalak t  a ttyUSB0 csom ponton
csatlakozik.) (devfs hasznÆlatakor usb/tts/0)
Generic converter now attached to ttyUSB1
(Az ÆltalÆnos Ætalak t  a ttyUSB1 ponton
csatlakozik.) (devfs hasznÆlatakor usb/tts/1)
```

Ebben az esetben a */dev/ttyUSB0* és a */dev/ttyUSB1* egyaránt használható adatcserére.
A végpontok sorrendje lényegtelen, akár azt is megtehetjük, hogy először felsoroljuk az összes bemenetet, majd a kimeneteket vesszük sorra. (Az előző példában váltakozva szerepeltek.)

Az USB-soros mag az összes ki- és bemeneti végpontot kezelni fogja, és az imént látott módon fogja őket párosítani. Minden tömeges adatmozgatásra szolgáló párhoz egy-egy megszakításvégpontot is rendel, ha létezik ilyen, de a megszakításvégpontot az általános illesztőprogram nem fogja használni, azt csak a rendszermag USB-soros illesztőprogramjai vehetik igénybe.
Ha az általános USB-soros illesztőprogramot hozzá szeretnénk kötni az eszközhöz, az eszköz USB-gyártóazonosítóját (USB vendor ID) és termékazonosítóját (product ID) betöltéskor átadott értékként kell az `usbserial` modul tudomására hozni. Ha például az imént látott eszközt `ffff` gyártóazonosítóval és `fff8` termékazonosítóval szeretnénk kötésbe hozni, az alábbi parancsot kell kiadni:

```
modprobe usbserial vendor=0xffff
product=0xfff8
```

Ha a felhasználótól nem várhatjuk el, hogy a megfelelő eszközazonosítóval betöltse az `usbserial` modult, vagy ha az USB-soros illesztőprogrammal egynél több eszközazonosítót is használni akarunk, akkor egy aprócska illesztőprogram írása a megoldás. Erre listánk (➲ http://www.linuxvilag.hu/USB) szolgáltat példát. Ebben az illesztőprogramban nincsenek visszahívó függvények, mindössze a vezérelni kívánt eszközök gyártó- és termékazonosítóit találjuk benne. Ennek alapján az `usb_serial_device_type` adatszerkezet megadása a lista 1-es szakaszában látható.
A megfelelő gyártó- és termékazonosítókat az `id_table` mutatóban kell felsorolni (a lista 2-es szakasza).
Az illesztőprogram mindössze két függvényt – egy két- és egy háromsorost –, valamint három változómegadást tartalmaz. Segítségével a megadott eszközökre vonatkozóan az általános USB-soros illesztőprogram összes szolgáltatását igénybe vehetjük. Az illesztőprogram betöltése önműködően lezajlik, amikor az eszközt csatlakoztatjuk a számítógéphez – azért ez sem rossz dolog. Alighanem ez az egyik legkisebb működő linuxos rendszermag-illesztőprogram. Fordítása az alábbi paranccsal történik:

```
echo "obj-m := tiny_tiny_usbserial.o"
➥> Makefile
make -C <a/rendszermag/elØrØsi/œtja>
➥ SUBDIRS=$PWD modules
```

A Windows USB OPOS-soros illesztőprogramon keresztül a Windows operációs rendszer is támogatja az ilyen jellegű, az eszköz számára virtuális „COM”-kapukat létrehozó eszközfelületet. Az eszközök fejlesztői, gyártói így olyan USB-s termékeket dobhatnak piacra, amelyekhez – szerencsés módon – nem kell külön linuxos és windowsos illesztőprogramot írni.

Az USB-soros eszközök életciklusa

Amikor egy USB-soros eszközt csatlakoztatunk a számítógéphez, különféle műveletek hosszú sora szükséges hozzá, hogy

Az „aprócska” USB-soros illesztőprogram

```
#include <linux/config.h>

#ifdef CONFIG_USB_SERIAL_DEBUG
       static int debug = 1;
       #define DEBUG
#else
       static int debug;
#endif

#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/tty.h>
#include <linux/module.h>
#include <linux/usb.h>
#include <../drivers/usb/serial/usb-serial.h>

#define TERM KAZONOS˝T     0xfff0
#define ESZK ZAZONOS˝T     0xfff0

/* 2-es szakasz */
static struct usb_device_id id_table [] = {
       {USB_DEVICE(TERM KAZONOS˝T ,
                  ➥ESZK ZAZONOS˝T )},
       { }    /* A bejegyzØs lezÆrÆsa */
};
/* 2-es szakasz vØge */

/*
 * az illesztıprogram  nmßk dı bet ltØsØnek
 * lehetıvØ tØtele
 */
MODULE_DEVICE_TABLE (usb, id_table);

static struct usb_driver tiny_driver = {
   .name =             "tiny",
   .probe =            usb_serial_probe,
   .disconnect =       usb_serial_disconnect,
   .id_table =         id_table,
};

/* 1-es szakasz */
/* Minden, a Tiny eszk z Æltal igØnyel adat */
static struct usb_serial_device_type  *
               ➥tiny_device = {
   .owner =            THIS_MODULE,
   .name =             "Tiny USB serial",
   .short_name =       "tiny",
   .id_table =         id_table,
   .num_interrupt_in = NUM_DONT_CARE,
   .num_bulk_in =      NUM_DONT_CARE,
   .num_bulk_out =     NUM_DONT_CARE,
   .num_ports =        1,
};
/* 1-es szakasz vØge */

static int __init tiny_init (void)
{
       usb_serial_register (&tiny_device);
       usb_register (&tiny_driver);
       return 0;
}

static void __exit tiny_exit (void)
{
       usb_deregister (&tiny_driver);
       usb_serial_deregister (&tiny_device);
}

module_init(tiny_init);
module_exit(tiny_exit);

MODULE_LICENSE("GPL");
```

adott USB-soros illesztőprogram egy tty-eszközt vezérelhessen. Ezek a műveletek a következők:

- Az USB-elosztó (hub) illesztőprogramja felismeri az új eszközt. Hozzárendel egy USB-azonosítót, kiolvassa belőle a fontosabb USB-leírókat, majd ezeket egy `usb_device` adatszerkezetbe helyezi, és kellő számú, a teljes USB-eszközt képviselő `usb_interfaces` adatszerkezettel párosítja.
- Az USB-mag gondozásába veszi az eszközt, és az USB-felületeket bejegyzi a rendszermag illesztőprogrammagjánál.
- A rendszermag illesztőprogrammagja végignézi a már bejegyzett USB-illesztőprogramok listáját, és megvizsgálja, hogy ezek valamelyike kezelni tudja-e az eszközt.
- Mivel USB-soros eszközről van szó, az USB-soros mag átveszi az eszköz vezérlését a rendszermag illesztőprogrammagjától.
- Az USB-soros mag `usb_serial` névvel egy adatszerkezetet hoz létre, majd ezzel az adatszerkezettel meghívja a megfelelő USB-soros illesztőprogram `probe()` függvényét.
- Az USB-soros illesztőprogram `probe()` függvénye – ha szükséges – üzembe helyezi (inicializálja) az eszközt, majd a vezérlést visszaadja az USB-soros magnak.
- Az USB-soros mag létrehozza az `usb_serial_port` adatszerkezeteket – az adott eszközön lévő soros kapuk számától függően –, majd meghívja az USB-soros illesztőprogram `attach()` függvényét, ha az rendelkezik ilyennel.
- Az `attach()` függvény visszatérését követően kerül sor az egyes `usb_serial_port` adatszerkezetek bejegyzésére a rendszermag illesztőprogrammagjánál.
- A rendszermag illesztőprogrammagja minden egyes kapuhoz egy visszahívást hajt végre az USB-soros mag felé.
- Az USB-soros mag meghívja a kapu USB-soros illesztőprogramjának egyedi `port_probe()` függvényét – ha van ilyen –, majd bejegyzi a kaput a tty-rétegnél, befejezve ezzel az üzembe helyezés folyamatát.

A folyamat lejátszódását követően a tty-eszközcsomópont az egyedi USB-soros kapuhoz kötődik. Amikor valamelyik felhasználó megnyitja az eszközcsomópontot , a rendszermagban az alábbi lépésekre kerül sor:

- A rendszermag megvizsgálja az eszközcsomópontot, és megállapítja, hogy azt a tty-réteg jegyezte be, tehát meghívja a tty-réteg `open` függvényét.
- A tty-réteg megvizsgálja az eszközt, és megállapítja, hogy az USB-soros mag jegyezte be nála a csomópontot, tehát a

drivers/usb/serial/usb-serial.c fájl `serial_open()` függvényét hívja meg.
- A `serial_open()` függvény meghatározza, hogy melyik egyedi USB-soros illesztőprogram van ehhez a csomóponthoz bejegyezve.
- A megadott USB-soros illesztőprogram modulszámlálóját a rendszer növeli, ezzel előzi meg a modul eltávolítását, miközben egy felhasználó esetleg éppen használja az eszközt.
- Ha a megadott USB-soros illesztőprogramnak van `open()` függvénye, akkor sor kerül ennek a meghívására, és átadott értékként a kapu `usb_serial_port` adatszerkezetét kapja meg.
- Az USB-soros illesztőprogram ekkor bármely szükséges, kifejezetten az eszközre jellemző megnyitó jellegű műveletet el tud végezni, és el tudja küldeni az eszköztől érkező adatok fogadásának megkezdéséhez szükséges URB-ket (USB-kérésblokkokat).

Ha egy felhasználó az eszközcsomópontra a `write()` függvényt hívja meg, vagyis a megadott soros kapura adatokat akar küldeni, a rendszermag az alábbi lépéseket hajtja végre:
- A rendszermag meghívja a tty-mag `tty_write()` függvényét. Korábban – az open híváskor – ezt a mutatót már beállította, így nem kell újra előkeresnie.
- A `tty_write()` meghívja a vonalszabályozó (line discipline) `write()` függvényét a megadott tty-eszközre.
- A vonalszabályozó meghívja az USB-soros mag `serial_write()` függvényét.
- A `serial_write()` függvény meghatározza, hogy pontosan melyik USB-soros illesztőprogramot használja ez a fájl, majd annak `write()` függvényét hívja meg.
- Az USB-soros illesztőprogram ekkor megkezdheti az adatoknak egy átmeneti tárba történő másolását, illetve az USB-kapcsolaton keresztül az eszköznek való elküldésüket, közben gondoskodik az eszköz által esetlegesen támasztott különleges formázási elvárások teljesítéséről.
- Miután az adatok küldése befejeződött, az illesztőprogram felébreszti a tty-eszközt, hogy az küldje el neki az esetlegesen átmeneti tárban tárolt adatokat. Mindez egy egyszerű hívással történik:

```
schedule_work(&port->work);
```

Ha adott kapu USB-soros illesztőprogramja adatokat kap, akkor a kapu átmeneti tárához (flip puffer) rendelt tty-adatszerkezetbe kell helyeznie őket:

```
for (i = 0; i < data_size; ++i) {
    if (tty->flip.count >= TTY_FLIPBUF_SIZE)
        tty_flip_buffer_push(tty);
    tty_insert_flip_char(tty, data[i], 0);
}
tty_flip_buffer_push(tty);
```

Amikor egy felhasználó az eszközcsomóponton `read()` hívást hajt végre, a kapu `tty flip` átmeneti tárában található adatokat kapja meg.
Ha a felhasználó lezárja az eszközcsomópontot, a rendszermag az alábbi lépéseket hajtja végre:
- A rendszermag meghívja a tty-mag `tty_release()` függvényét.
- A `tty_release()` meghatározza, hogy ez volt-e az utolsó hivatkozás erre az eszközcsomópontra. (Ne feledjük, egy eszközcsomópontot egyszerre több program is megnyithat.) Ha igen, az USB-soros mag `serial_close()` függvényének meghívására is sor kerül.
- A `serial_close()` függvény meghívja az USB-soros illesztőprogram `close()` függvényét, így lezárulnak a függőben lévő USB-átvitelek, az illesztőprogram pedig készenléti állapotba kerül.
- Az USB-soros mag csökkenti az USB-soros illesztőprogram modulszámlálóját, ami akár el is távolítható.

Az USB-soros eszközök megjelenése sysfs alatt

Az USB-soros eszköz USB-soros illesztőprogramhoz való kötésének fenti leírásában többször is sor került a rendszermag illesztőprogrammagjának meghívására. Erre azért van szükség, mert az USB-soros mag külön buszként jelenik meg a rendszermag illesztőprogram-modelljében – így egyetlen USB-eszközön több kapu is lehet.
Például az alábbi egy nyolckapus USB-soros eszköz, ami a rendszer első USB-buszán található. Sysfs alatt az elérési útja a következő: */sys/devices/pci0/00:09.0/usb1/1-1/1-1.1*. Ezen a könyvtáron belül az alábbi könyvtárakat és fájlokat találjuk: ***1-1.1:0/, bcdDevice, bConfigurationValue, bDeviceClass, bDeviceProtocol, bDeviceSubClass, bmAttributes, bMaxPower, bNumConfigurations, bNumInterfaces, idProduct, idVendor, manufacturer, name, power, product, serial, speed, ttyUSB0/, ttyUSB1/, ttyUSB2/, ttyUSB3/, ttyUSB4/, ttyUSB5/, ttyUSB6/*** és ***ttyUSB7/***.
A könyvtár állományaiból minden, az eszközzel kapcsolatos adatot megtudhatunk, ahogy az ***1-1.1:0/*** könyvtár állományaiból is, ez egyébként az első kapcsolódási felület az eszköz felé. A `ttyUSB*` könyvtárakat az USB-soros mag hozza létre, és az alábbi fájlokat tartalmazzák: ***dev***, ***name*** és ***power***.
A ***dev*** fájl az adott eszköz al- és főszámát tartalmazza; ha párbeszédet akarunk folytatni az eszközzel, ennek alapján található meg pontosan a megfelelő eszközcsomópont. A */sys/bus/usb* könyvtár szerint ez az USB-eszköz az `io_edgeport` USB-illesztőprogramhoz kötődik. Van egy `usb-serial` busz is, ami a rendszermagnál bejegyzett egyes USB-soros kapukat mutatja. Mivel ezek a kapuk tty-eszközök, a tty-osztály könyvtárában is megjelennek.
Mivel a hivatkozások mindegyike ugyanahhoz az USB-eszközhöz vezet, az USB-eszköz típusa, tty-kapuinak a száma és a vezérléséhez szükséges USB-soros illesztőprogram típusa könnyedén meghatározható. Így sokkal több adathoz juthatunk hozzá, mint amit a */proc/tty/driver/usb-serial* fájlban találtunk, lásd írásom első részében.
A `sysfs` felülettel kapcsolatban most szűkszavú leszek, ám érdemes tudni, hogy rengeteg adatot szolgáltat a rendszerben pillanatnyilag jelen lévő fizikai és logikai eszközökről. A `sysfs`-nek és a rendszermag illesztőprogram-modelljének bővebb ismertetését ***Pat Mochel*** idei ***linux.conf.au*** írásában, a ➲ http://www.kernel.org/pub/linux/kernel/people/mochel/doc/lca címen találod meg.

Linux Journal 2003. április, 108. szám

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

Csatlakozás a Hálóra Linuxszal (2. rész)

Az előző részben összeállítottunk egy működő útválasztót a Coyote Linux-változat segítségével. Ebben a részben az útválasztó további beállításairól, távoli karbantartásáról és bővítéséről lesz szó.

Először is nem árt ismerni az útválasztó működési elvét. Mivel a szolgáltatótól csak egy valós (Interneten használható) IP-címet kapunk, belső hálózatunk gépei „saját nevükben" nem tudnak az Internetre csomagot küldeni és onnan fogadni. Az útválasztó ezt úgy oldja meg, hogy az úgynevezett NAT (Network Address Translation: hálózati címváltás) egyik módszerét használja: az IP-álcázást (masquerading). Az álcázás úgy valósul meg, hogy a belső hálózat gépei által küldött csomagokat az útválasztó a saját IP-címével egy eltérő kapuról küldi ki az Internetre, és ugyanígy fogadja a választ, majd a belső IP-címet visszahelyezve továbbítja azt a belső hálózat gépére. Egy táblázatot tart fenn annak érdekében, hogy ezeket az IP-cím-átváltásokat el tudja végezni. A belső gépek úgy látják, mintha közvetlenül az Interneten lévő géppel lennének kapcsolatban, az Internet felől pedig úgy tűnik, mintha csak az útválasztóval tartanának kapcsolatot. Itt hívnám fel a figyelmet rá, hogy elvben megközelítőleg megállapíthatjuk, hogy hány gép működik az útválasztó mögött „rejtve" (lásd ***Steven M. Bellovin*** cikkét). Ezért, ha a szolgáltatónk kifejezetten tiltja a több géppel történő használatot, akkor, ha meg is építjük az útválasztót, ne nagyon használjuk több géppel egy időben (csak az Internetre irányuló forgalom számít). Én azon az állásponton vagyok, hogy ha a felhasználási szerződés kifejezetten nem tiltja, akkor szabad a pálya. (Mivel nincs benne, hogy csak egy adott operációs rendszerrel lehet igénybe venni, ugyanekkora terhelést el lehetne érni egy linuxos, illetve unixos gépen, és arra a helyi hálózaton bejelentkező felhasználók esetén is.)

Hibalehetőségek a telepítés során

Biztosan többféle hiba merülhet fel a telepítés után. Egy elég részletes GYK (FAQ) található meg hosszas keresés után a ➲ http://www.dalantech.com oldalain. Itt egyébként rendkívül széles körű, igaz, angol nyelvű technikai adathalmaz található. Egy hibát említenék, ami elsőre eléggé nehezen volt felfedezhető. A modemes behívásnál egy úgynevezett chatprogram veszi fel a kapcsolatot először a szolgáltató gépével. Miután ez befejezte, akkor indul el a `ppp` program; sok esetben ez a program kezeli a bejelentkezést is. Ennek kérését a kiszolgáló általában egy jelszó-készenlétijel (`login:` prompt) küldésével jelzi. A mi esetünkben azonban ehelyett `username` készenléti jelet küldött, így a chatprogram hiába várt a jelszóablakra – ezt tehát módosítani kellett.

Több módszer is akad a Coyote rendszer módosítására. Az egyik, hogy billentyűzetet és monitort csatlakoztatunk a géphez, és bejelentkezünk. Lényegében ugyanez a módszer akkor is, ha `ssh` segítségével már megy a távoli bejelentkezés. Csak egyetlen felhasználó van: a rendszergazda. Az első bejelentkezéskor adhatunk meg jelszót. Bejelentkezés után az `lrcfg` program indul el. Ebből a `q` ponttal léphetünk ki. Ha nem tudjuk, melyik könyvtárban vagyunk (a rendszer nem írja ki a készenléti jelnél), a `pwd` paranccsal megtudhatjuk. A fent említett hiba kiküszöböléséhez lépjünk a */etc/ppp* könyvtárba, és itt szerkesszük a ***szolgaltatonev.chat*** fájlt. (A szolgáltatónevet a lemez készítése közben mi adtuk meg. Ezért kell rövidnek és ékezet nélkülinek lenni.) A Coyote rendszernek része az `ae` szövegszerkesztő. Ezt az `ae fÆjlnØv` paranccsal elindítva elvégezhetjük a szükséges módosításokat. Az F1 lenyomásával kaphatunk segítséget a billentyűparancsokhoz. Ebben az esetben az `ogin:` részt kell kicserélni a `sername:` bejegyzésre. (Azért nincsenek ott az első betűk, mert nem lehet tudni, hogy kis- vagy nagybetűvel kezdve küldi-e a kiszolgáló.)

Miután módosítottunk a rendszeren, tudnunk kell, hogy a módosítás csak a memórialemezen következett be, azaz újraindítás esetén elvész, ezért mielőtt újraindítanánk, hajlékonylemezre kell mentenünk. Ha átkapcsoltuk az írást tiltó fület, akkor újra engedélyezzük. (A végső beállítás után nem árt csak olvashatóvá tenni, így egy esetleges betörés esetén se tudnak olyan kárt tenni az útválasztóban, amit egy újraindítás ki ne küszöbölne.) Indítsuk el a `lrcfg` programot. Ebben válasszuk a ***b*** pontot. A mentés hamarosan elkészül. Ezek után újabb indításkor már a módosított állományt használja a gép.

Pár szó a Coyote felépítéséről

A különböző feladatokhoz szükséges állományokat tgz- (***tar.gz***) fájlokban, tömörítve tárolja a hajlékonylemezen. Ezzel (is) magyarázható az, hogy ennyi minden elfér egyetlen lemezen. Indításkor önműködően létrehoz egy memórialemezt, és ebbe csomagolja ki a szükséges állományokat – így alakul ki a működő fájlrendszer.

Ebből adódik a másik módosítási lehetőség. Az útválasztó lemezét saját linuxos munkaállomásunkon is módosíthatjuk. Fűzzük be a hajlékonylemezt a rendszerbe, majd másoljuk a megfelelő ***.tgz***-fájlt (a fenti feladatnál ez az ***etc.tgz*** volt) egy üres könyvtárba, ott csomagoljuk ki a `tar xvzpf fÆjlnØv` paranccsal. Ezután a létrejövő könyvtárakban és fájlokban elvégezhetjük a szükséges módosításokat; utána a `tar cvzpf fÆjlnØv` paranccsal csomagoljuk vissza az állományokat, majd másoljuk vissza őket a lemezre.

Egy jó tanács: ha működő rendszerünk van, akkor a módosítgatások előtt célszerű lenyomatot készíteni róla, hogy ha valami rosszul sülne el, pótlemezt készíthessünk. A szükséges lenyomat a

```
dd if=/dev/fd0u1680 of=lenyomatnev.fÆjl
```

paranccsal készíthető el. Ebből hajlékonylemezt az `if` és `of` kapcsolók cseréjével készíthetünk.

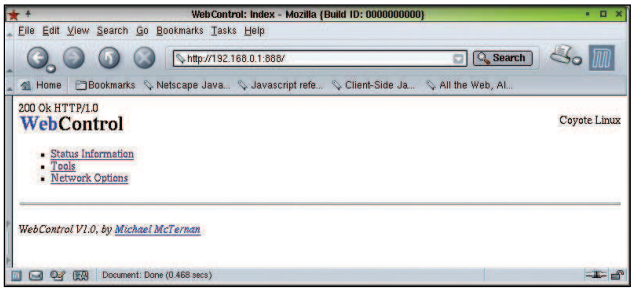
Távoli karbantartás

Előfordulhat, hogy a munkaállomásokon is tudnunk kell az útválasztó IP-címét. Az egyik kézenfekvő példa erre az, ha azt szeretnénk, hogy valamelyik (feltehetően Linuxot futtató) szeretnénk,

hogy valamelyik (feltehetően Linuxot futtató) munkaállomás webkiszolgálóként működjön, és így lehetővé váljon, hogy bár változó IP-címmel, de teljes dinamikus valójában megmutathassuk webes munkánkat másoknak. Ehhez azonban meg kell tudnunk mondani, hogy az útválasztó éppen milyen IP-címen érhető el, és emellett a kívánt HTTP-kaput át kell irányítanunk az adott munkaállomásra (erről az utóbbiról később lesz szó).
Ha telepítettük a `webctl` csomagot, akkor viszonylag könnyen megtudhatjuk a hálózati adatokat. Írjuk be a böngészőbe (Mozilla ajánlott, a régebbi Netscape-pel gondok akadtak a megjelenítéssel során):

192.168.0.1:888

Azért nem a szabványos 80-as kapun működik, hogy azt át tudjuk irányítani. Ekkor a *képen* látható egyszerű felület jelenik meg.



Innen a ***Status Information*** hivatkozásra (link) kattintva részletes tájékoztató oldalt kapunk, gyakorlatilag az `ifconfig`, `route`, és az `ipchains -L` parancsok webesített változatát. A ***Tools*** hivatkozást követve a naplófájlok tartalmát nézhetjük meg, valamint egy egyszerű névfeloldást használhatunk (DNS lookup). A ***Network options***-t választva engedélyezhetjük a `telnet`-elérést, tilthatunk, illetve megnézhetjük a tűzfal szabályait. Elvileg hozzá is lehet adni, de ez a lehetőség egyelőre nem működik.
A másik lehetőség a távoli karbantartásra az SSH alkalmazása. Ha a Coyote-lemez elkészítésekor kiválasztottuk ezt a csomagot, akkor fent van az útválasztón. (Az útválasztóra bejelentkezve és a `ps` parancsot kiadva egy */usr/sbin/sshd* folyamatot kell látnunk). Ahhoz azonban, hogy egy munkaállomásról be tudjunk lépni, egy kulcspárt kell készítenünk, és a nyilvános részét fel kell telepítenünk a Coyote-ot futtató gépre. Figyelem, ha már van `ssh` kulcspárunk (mert más célra készítettünk már egyet), és használjuk is, akkor az alábbi parancsokat ne futtassuk alapértelmezett módon, mert felülírja a régi kulcsot. Ha régebbi, 1-es változatú SSH-nk van, akkor elég az alábbi parancsot a saját könyvtárunkban kiadni:

```
ssh-keygen
```

Ha a Linuxunk frissebb, és már van rajta SSH2, akkor meg kell adnunk azt is, hogy milyen fajta kulcsokat készítsen. Mivel a Coyote SSH1-t használ, így ennek megfelelő kulcsot készítünk az alábbi paranccsal:

```
ssh-keygen -t rsa1
```

Megkérdezi, hogy hová mentse a kulcsot – ha nincs másik kulcsunk, hagyjuk az alapértelmezett értéken. Ezután (kétszer) be kell írnunk egy jelszót, ami a kulcsunkat védi, hogyha valaki le tudja is másolni a gépünkről, akkor se tudja enélkül használni. Célszerű tehát beírni valamit (üres kulcsot is megenged). A rendszerjelszóval ellentétben szóközt is tartalmazhat, és hosszabb szöveg is begépelhető.
Ha elkészült, akkor nincs más dolgunk, mint átmásolni az útválasztóra. Mivel az SSH még nem működik, ilyen módon vagy a fentebb említett rendszerlemez teljes Linux-rendszerünkhöz való csatlakoztatásának módszerét alkalmazhatjuk, vagy pedig egyszerűen másoljuk ki a kulcs nyilvános felét tartalmazó *~/.ssh/identity.pub* fájlt egy hajlékonylemezre. Ezután a lemezt a Coyote rendszeren az alábbi parancsokkal csatlakoztassuk, és másoljuk be a kulcsot a megfelelő helyre:

```
cd /
mkdir /floppy
(tegy k be a lemezt a kulccsal)
mount /dev/fd0 /floppy
mkdir ~/.ssh
cp /floppy/identity.pub ~/.ssh/authorized_keys
umount /floppy
```

Ezek után, ha minden jól ment, készen állunk a próbára. A munkaállomáson, amelyről a kulcsot vittük, a következő paranccsal próbálkozzunk:

```
ssh 192.168.0.1
```

Megkérdi, hogy felvegye-e a 192.168.0.1 kiszolgálót az ismert gépek (known hosts) listájára. A teljes `yes` szó beírásával kell válaszolnunk. Ezután a jelszót (passphrase) kéri, írjuk is be.
Ha minden jól ment, azon nyomban látjuk az `lrcfg` program menüjét, és benn is vagyunk az útválasztón. Még az van hátra, hogy az új beállítást mentsük. Ehhez előbb módosítanunk kell a */etc/coyote/packages* fájlt. Írjuk bele, hogy a `root` csomagot is mentse (a `root` szót tartalmazó sor). Ezt követően az `lrcfg` programot elindítva és a ***b*** menüpontot választva menti a rendszert, és a következő indításkor már ezt használja. Ha elkészült, a fenti fájlt visszaírja az eredeti állapotra (a `root` sort kivéve), és újra menti a rendszert – a `root` csomagot a további mentésekből kizárhatjuk, és így jelentősen felgyorsíthatjuk őket.
Itt mindent ugyanúgy meg tudunk tenni, mintha közvetlenül az útválasztó billentyűzetét és monitorát használva jelentkeztünk volna be. Ha működik, nincs is szükség rájuk, le is vehetjük őket. (Ha esetleg nagyot hibázunk és elbabrálunk valamit, akkor lesz rájuk újra szükségünk.) Próbaként lefuttathatjuk az `ifconfig`, a `route` és a `dmesg` parancsokat – ezekből sokat megtudhatunk a rendszer állapotáról.

Az útválasztó finomhangolása

A rengeteg lehetőség közül kettőt ragadok ki. Az egyik a kapuváltás, hogy a külvilág számára egy belső munkaállomás valamilyen kiszolgálóként működhessen. A másik pedig több belső hálózat kezelése.
A kapuátirányítás elsőre rendkívül könnyűnek tűnik, mivel egy erre beépített rész található az IP-álcázás parancsfájljában. Ez azonban alaphelyzetben nem működik, sajnos bele kell javítanunk (valószínűleg változott a különböző változatok között).
A kívánt kapuváltásokat a */etc/coyote/portforwards* fájlban kell megadnunk, egy sorban egyet, a következő formában: kezdőkapu végsőkapu protokoll célgép.
Ha egy sort kettős kereszttel (#) kezdünk, azt megjegyzésként kihagyja. A különböző adatok között szóköznek és nem veszszőnek kell lennie. Ha a HTTP-kaput akarjuk egy belső gépre átirányítani, használhatjuk a következő sort:

```
80 80 tcp 192.168.0.5
```

A protokoll lehet UDP is, ha arra van szükség. Egy kapuprotokollpárt csak egy belső gépre lehet átirányítani.
Most van soron a parancsfájl javítása. Nyissuk meg a `/etc/rc.d/rc.masquerade` fájlt. A `set_port_forward` függvényt módosítsuk a következő formára:

```
set_port_forward() {
if ! [$# = 4] ; then
    echo " rvØnytelen kapuvÆltÆsi szabÆly a
    ➥/etc/coyote/portforwards fÆjlban!"
    return 1
    fi
    /sbin/ipmasqadm autofw -A -r $3 $1 $2
    ➥-h $4
    }
```

Ha ezek után le is futtatjuk a parancsfájlt (`/etc/rc.d/rc.masquerade`), akkor egy külső, Interneten lévő gépről ki is próbálhatjuk, hogy meg tudja-e nyitni a belső gépen lévő HTTP-kiszolgáló nyitólapját. Ha működik, akkor a szokott módon mentsük a rendszert (`lrcfg` ***b*** pont).
Most pedig bővítsük ki az útválasztót két belső hálózat kiszolgálására. Mindkettő számára biztosítja az IP-álcázást az internetéréshez, emellett – mint egy hagyományos útválasztó – össze is köti a két hálózatot, vagy ha a hálózati hordozó eltérő (például koax és sodrott érpár), akkor hídként. További szerepe lehet még a hálózat fizikai határának kitolása: tőle mindkét irányban a legnagyobb kábelhosszt lehet alkalmazni, így a két legtávolabbi gép az eredetileg lehetséges távolság kétszeresére lehet egymástól.
Ha a fenti példához hasonlóan a második belső hálózathoz ugyanolyan típusú hálózati csatolót alkalmazunk, mint az elsőhöz vagy mint az internetoldalihoz, és emellett a kártya PCI-sínes, akkor semmit sem kell tennünk az eszköz felismeréséhez, mert a modul az összes azonos típusú kártyát önműködően kezeli. Ha másfajta kártyát használunk, akkor a megfelelő modulokat a Coyote rendszer */lib/modules* könyvtárába kell telepíteni, majd a */etc/modules* fájlba kell beírni őket, ha szükséges, az értékekkel együtt. Ezután pedig az előzőekhez hasonlóan menteni kell a rendszert.
Újraindításkor figyeljük meg, hogy az összes kártya elindult-e. Ha igen, akkor már csak a program beállítása maradt hátra. Mivel ehhez nincs külön parancsfájl, kézzel kell megtennünk. Először is módosítsuk a */etc/coyote/coyote.conf* fájl hálózati adatokkal kapcsolatos részét az alábbiak szerint:

```
LOCAL_IPADDR=192.168.0.1
LOCAL_NETMASK=255.255.255.0
LOCAL_BROADCAST=192.168.0.255
LOCAL_NETWORK=192.168.0.0
LOCAL_IPADDR2=192.168.1.1
LOCAL_NETMASK2=255.255.255.0
LOCAL_BROADCAST2=192.168.1.255
LOCAL_NETWORK2=192.168.1.0
```

Mint látható, egy második hálózati beállításcsomagot adtunk hozzá, és négy újabb meghatározást (ezek egyébként a héjváltozók). Ettől még nem működik a második hálózat; azért van rá szükség, hogy könnyebben be tudjuk írni a megfelelő értékeket a parancsfájlokba.
Először is állítsuk be a `/etc/rc.d/rc.inet` parancsfájlt. Ez indítja el a hálózati eszközök kezelőprogramját. Írjuk be a kisebb jellel (>) jelölt sorokat a jelöletlen sorok közé. (A > jel nélkül).

```
....
# Set the interfaces to use for Internet /
➥Internal Networks
if [ -z "$IF_LOCAL" ]; then
       IF_LOCAL=eth0
fi
> if [ -z "$IF_LOCAL2" ]; then
>      IF_LOCAL2=eth2
> fi
.....
# Configure the local network
ifconfig $IF_LOCAL $LOCAL_IPADDR netmask
➥$LOCAL_NETMASK broadcast $LOCAL_BROADCAST
if [ $? = 0 ]; then
       LOCAL_UP=YES
else
       LOCAL_UP=NO
fi

> ifconfig $IF_LOCAL2 $LOCAL_IPADDR2 netmask
➥$LOCAL_NETMASK2 broadcast $LOCAL_BROADCAST2

# Enable kernel level IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

...
```

Az `ifconfig` egy sorban helyezkedik el a `$LOCAL_BROADCAST-tal`. Ezzel a hálózati csatolók egy újraindítás után elindulnak, és mivel a magban engedélyezett az IP-csomagtovábbítás, a két hálózat között máris hídként működik. A második hálózat internetéréséhez még bele kell írnunk az `rc.masquerade` parancsfájlt. A következő sort a hasonló első hálózati csatolóhoz tartozó sor után írjuk be (egyetlen sorba gépelve):

```
/sbin/ipchains -A forward -j MASQ -s
➥$LOCAL_NETWORK2/$LOCAL_NETMASK2 -d 0.0.0.0/0
```

Ezzel készen is vagyunk. Ha nem hibáztunk, újraindítás után (vagy a megfelelő parancsfájlok végrehajtása után) mindkét hálózat internetelérése, valamint a két hálózat közötti forgalom is működni fog.

Összegzés

Mint láthattuk, egy kisméretű Linux-terjesztés segítségével könnyedén megoldhatjuk helyi hálózatunk internetelérését. Az eszközköltség rendkívül csekély, mivel az útválasztót régi és használt gépalkatrészekből is összeállíthatjuk. A rendszer létrehozása és beállítása nem bonyolult, kevés karbantartást igényel és üzembiztos.

Havránek Ferenc
Automatikamérnökként dolgozik. Kedvtelései közé tartozik mindenféle kétkerekű járművön (kerékpár és motor) való közlekedés. Ezenkívül szívesen tölti idejét programozással, nemcsak PC-s, hanem egyéb környezetben is, például mikrovezérlő programokat ír.

Anyagmintázatok a Blenderben (4. rész)

Ebben a részben szeretném bemutatni a Blenderben használható felületi mintázatok főbb típusait és azok jellemzőit.

Ahhoz, hogy egy tárgy mintázatát elkészíthessük, először az anyagát kell meghatároznunk – ennek módját sorozatunk előző részében ismerhették meg. A felületi minták elkészítéséhez – miután a tárgyat kijelöltük – kapcsoljunk az F6 billentyűvel a mintázatszerkesztő nézetre. Alapállapotban a tárgyaknak még nincsen mintázatuk, tehát első lépésben új mintázatot kell létrehoznunk. Az új anyag létrehozásához hasonlóan ebben a nézetben is találhatunk egy legördülő menüt a nézet fejlécének közepén, ha egy vízszintes vonalkához hasonló ikonra kattintunk. Itt láthatjuk majd az elkészült mintázatokat, azonban kezdetben a menüben csak az ***ADD NEW*** pont szerepel. Pontosan erre van most szükségünk, tehát aktiváljuk ezt a menüpontot. Ennek hatására a fejléc alatt egy újabb gombsor jelenik meg, a segítségével beállíthatjuk majd a mintázat tulajdonságait.

A mintázat tulajdonságainak beállítása

Kezdjük az ismerkedést az egyszerűbb mintákkal, hiszen ezeket megismerve tudásunkat a későbbiekben is jól hasznosíthatjuk. Kattintsunk tehát a ***Clouds*** gombra. A nézetben megjelenő újabb értékeket megváltoztatva alakíthatjuk ki a felhőmintázat végleges formáját, az alapvető tulajdonság a ***Noise Size***. A mintázatok nagy többségének létrehozása során a Blender egy rekurzív (önmagát újra és újra meghívó) algoritmust használ, ekkor a mintázatot alkotó kép négy sarkába egy-egy képpontot helyez el, majd ezt az eljárást ismétli oly módon, hogy a képet négy negyedre osztja fel, és a következő képpont értékét az előző méret sarkainak értékéből számítja ki. Az így kapott értéket tetszőleges véletlen számmal növeli vagy csökkenti, hogy a mintázat ne legyen teljesen szabályos. Ezt a véletlen számot nevezzük a kép létrehozása során zajnak, és mértékét a ***Noise Size*** kapcsolóval határozhatjuk meg. Az ilyen képek alkotásakor látványos változásokat érhetünk el, ha nem minden újonnan kiszámított képpontnál adunk zajt a kapott értékhez. A ***Noise Depth*** arra szolgál, hogy beállíthassuk a képpont értéke változásának gyakoriságát. Az alapvető értékek alatt még két kapcsolót láthatunk: a ***Soft Noise***-t, ezzel finomabb mintázat készíthető, míg a ***Hard Noise***-zal durvább minták valósíthatók meg.

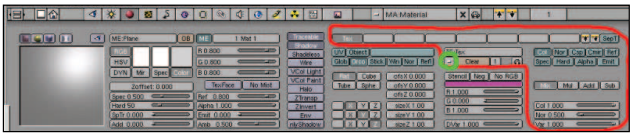
A mintázat színárnyalatának megadása

A mintázatszerkesztő nézet jobb oldalán található a ***Colorband*** feliratú kapcsoló, amivel azt állíthatjuk be, hogy a mintázatot milyen színárnyalatok alkossák. A színárnyalatok beállításakor megadhatjuk az átmenetet képező színeket, azok átlátszóságát és minden létrehozott összetevő helyzetét a teljes átmenet szélességén belül. Új színeket adhatunk az átmenethez az ***Add*** gomb használatával, a már meglévőket pedig a ***Del*** gomb alkalmazásával törölhetjük. E két gomb között találhatjuk a pillanatnyi szín sorszámát, meg is tudjuk változtatni, és a továbbiakban a változtatások (színösszetevők, átlátszóság stb.) a jelenlegi összetevőre vonatkoznak majd. A színátmenet beállításai alatt találhatjuk még a teljes mintázat világosságát és kontrasztját meghatározó értékeket, amelyeket a szokott módon, a gomb közepétől balra vagy jobbra kattintással, vagy – a lenyomva tartott SHIFT billentyűvel együtt történő kattintással – közvetlenül adhatunk meg.

Az egyszerű mintázatok alaptulajdonságaival megismerkedve a továbbiakban csak a különbségekkel és az új tulajdonságokkal kell bővebben foglalkoznunk; ezekkel azonnal találkozhatunk, amikor a ***Clouds*** mellett található ***Wood*** mintázatot vesszük szemügyre.

Wood

Ezt a mintát használhatjuk például valamilyen fából készült tárgy textúrájaként, de alkalmazási lehetőségeinek csak fantáziánk szabhat határt. Ha jobban belegondolunk, akár egy



A mintázatok beállításai

zebracsíkos takaró anyagát is létrehozhatjuk faerezet mintájára. Ennél a mintázatnál kétféle elrendezésben lehetnek a különféle csíkok, attól függően, hogy a kiindulásul szolgáló fatörzset keresztben vagy hosszában vágjuk-e ketté. Amikor hosszanti metszetet készítünk, használhatjuk a ***Bands*** és a ***BandNoise*** gombokat, a törzs keresztmetszetének létrehozására pedig a ***Rings*** és a ***RingNoise*** típusok alkalmasak. A Noise angol szó jelentése zaj, vagyis a ***Bands*** és a ***Rings*** a zaj nélküli változatokat jelölik, amelyek szabályos mintát adnak, míg a ***BandNoise*** és a ***RingNoise*** a mesterségesen szabálytalan mintázat megjelenését eredményezi, mivel ezeket a Blender véletlenszerűen módosítja. A típus meghatározása után beállíthatjuk még a zaj mértékét és a ***Turbulence*** érték segítségével egyfajta örvényszerű változást hozhatunk létre a mintázatban. Természetesen itt is alkalmazhatunk lágyabb vagy erősebb véletlen zajkeltést, a szokott helyen található ***Soft Noise*** és ***Hard Noise*** kapcsolókkal.

Marble

A következő beépített mintázat a ***Marble***, ami beállításaiban is nagyon hasonlít az előző mintához, de márványszerű felületek kialakítására alkalmas. Itt csak a ***Soft***, a ***Sharp*** és a ***Sharper*** kapcsolókra hívnám fel a figyelmet, amikkel az átmenetek finomságát szabályozhatjuk. A többi kapcsoló a fentiek alapján már ismert, így vegyük sorra a következő mintázatot.

Magic

A Magic mintát valójában igen változatos hatások létrehozására használhatjuk, még annak ellenére is, hogy szavakkal ezeknek a változatos mintáknak a leírása csak nehezen adható vissza. Gyakorlatilag arról van szó, hogy egy kétváltozós szinuszos és koszinuszos tagokkal leírható függvény értékeit változtatjuk.

Természetesen itt is használhatunk örvényszerű véletlen módosítást a látvány fokozására. Ne feledkezzünk meg a jobb alsó részen található vörös, zöld és kék összetevők módosításáról sem, hiszen a segítségükkel fedezhetjük fel a varázslatos mintában rejlő végtelen lehetőségeket.

Mintázatok színátmenetének és felületének meghatározása

A következő beépített mintázat egyszerű színátmenetek létrehozására alkalmas, de természetesen itt is meghatározhatjuk az átmenet kialakításának a típusát. Létrehozhatunk lineárisan vagy négyzetesen változó átmenetet, átlós átmenetet, valamint sugárirányú átmenetet. A sugárirányú színátmenet létrehozása során is megadhatjuk, hogy egyenletesen vagy négyzetesen változzon-e a szín a megadott határokon belül. Erre alkalmas a ***Sphere*** és a ***Halo*** változat.
Érdes felületek – például falak – mintázatának létrehozására megfelelő a ***Stucci*** típus és ennek három változata. A ***Plastic*** finom átmeneteket képez a mintázatban, míg a ***Wall in*** és a ***Wall out*** durvább felületi mintázat kialakítására alkalmas. A ***Wall out*** típus szintén durvább mintázatot eredményez, de az előbbi negatív képét használva.
Az utolsó egyszerű mintázat a Blenderben a ***Noise,*** az előzőekben már ismertetett általános kapcsolókkal.

A mintázatok tárgyhoz rendelése

Mindezek után már képesek vagyunk egy-egy tárgy anyagának elkészítésére és mintázatokat is tudunk alkotni. Tekintsük át, hogy a létrehozott mintázatokat milyen módon tudjuk a tárgyakhoz rendelni, hogyan tudjuk a mintázat által a tárgynak nemcsak a színét módosítani, hanem a felület normálvektorait és a tárgy fényvisszaverő képességét egyaránt, illetve miként tudunk több mintázatot rendelni egy tárgyhoz.
Kapcsoljunk vissza az anyagtulajdonságok beállítására szolgáló nézetbe az F5 billentyű használatával. Ennek a nézetnek a jobb oldalán találhatjuk azokat a kapcsolókat és értékeket, amelyekkel az anyag mintázatát határozhatjuk meg. A mintázatot tehát nem a tárgyhoz rendeljük hozzá, hanem az anyagokhoz, így amelyik tárgyon látni szeretnénk a mintázatot, annak az anyagához kell hozzárendelünk. Alapállapotban, amikor az anyagtulajdonságok nézetéből átváltottunk a felületi mintázatok beállításához, már a jelenlegi anyag mintázatát hoztuk létre és kezdtük módosítani. Ez azt jelenti, hogy amennyiben csak egy mintázatot szeretnénk a tárgyon viszontlátni, akkor a továbbiakban nincs semmi különleges tennivalónk.

Egy anyaghoz több mintázat

Ha azonban azt szeretnénk, hogy az anyaghoz több mintázat is tartozzon, például az egyik meghatározza majd a színek változását, a másik pedig a felületi egyenetlenségeket, akkor nézzük meg egy kicsit jobban az anyagtulajdonságok nézetét!
Képünkön pirossal jelölve látható az a terület, amire nagyobb figyelmet kell fordítani.
A bejelölt terület felső gombsorán a pillanatnyi mintázatot választhatjuk ki. Itt látható, hogy alapállapotban az elsőként létrehozott mintázat hozzá van rendelve az anyaghoz. A jelölt terület jobb oldalán találhatjuk azokat a kapcsolókat, amelyek meghatározzák, hogy az anyag mely tulajdonságát módosítsa a felületi mintázat. A ***Col*** használatával az anyag színe változik, a ***Nor***-éval a mintázatban megjelenő világosságértékek a normálvektorokra lesznek hatással, a ***Csp*** alkalmazásával a tárgy fényes felületeinek színét módosítja a mintázat, a ***Cmir*** hatására a tükröződések színe változik, a ***Ref*** bekapcsolásával pedig az anyag fényvisszaverő képességére hat majd a mintázat.

Külön figyelmébe ajánlom mindenkinek a normálvektorok módosításában rejlő lehetőségeket, hiszen így olyan módon hozhatunk létre érdes felületeket, hogy a tárgyat alkotó háromszögek száma nem változik meg. E kapcsolók alatt találhatjuk a ***Hard*** kapcsolót, amivel az anyag keménysége befolyásolható, a ***Alpha*** az átlátszóságot befolyásolja, míg a ***Emit*** az anyag sajátfény kibocsátását módosítja a mintázat intenzitásértékeinek megfelelően.

Rétegműveletek

Lefelé haladva négy újabb kapcsolót találhatunk, amelyek azt határozzák meg, hogy milyen műveletet végezzen a Blender a mintázat egyes rétegei, vagy ha csak egy textúrát használunk, akkor a mintázat és az anyag színe és egyéb tulajdonságai között. Így a ***Mix*** egyszerűen csak módosítja az anyagot vagy a többi réteget, a ***Mul*** szorzásműveletet végez a rétegek között, a ***Add*** összead, míg a ***Sub*** pedig kivonja egymásból a rétegeket. Ezeknek a kapcsolóknak a hatása igazából kísérletezéssel tapasztalható meg, hiszen a Blender jobban tudja, hogy pontosan milyen értékek vannak a mintázatban, és a műveletek eredményét is könnyebben megítélhetjük, ha nem nekünk kell az értékeket kiszámítgatnunk. Irányadóként elmondható, hogy pontos tervezés, számolás helyett inkább vegyük a mintázat és az anyag általános világosságát, és ennek alapján gondoljuk végig a lehetséges hatást. Ha például egy világos anyagra kivonás művelettel helyezek rá egy sötét mintázatot, akkor az anyag csak kevéssel lesz sötétebb, mintha ugyanezt a mintázatot az összeadás műveletét alkalmazva helyezem a tárgyra – ekkor világosabb végeredményt kapok.
Az anyagtulajdonságok meghatározására szolgáló nézet jobb alsó sarkában még három fontos vezérlőt találunk. Ez a ***Col***, ami azt határozza meg, hogy az adott mintázat milyen mértékben változtassa meg az anyag színét, a ***Nor***, amellyel a normálvektorokra gyakorolt hatás mértéke adható meg, és a ***Val***, amivel az anyag egyéb tulajdonságaira (például átlátszóság, saját fény) gyakorolt hatás szabályozható. Ezek az értékek százalékban értendők, és arra alkalmasak, hogy az anyag tulajdonságait finomhangoljuk.
További mintázatokat úgy rendelhetünk az anyaghoz, hogy a képen a pirossal jelölt terület felső részén található gombok közül egy másikat választunk ki, és a zölddel jelölt legördülő menüből kiválasztjuk a kívánt mintázatot. Természetesen minden felületi mintázathoz beállíthatók a fent ismertetett értékek.

Összegzés

Ezzel áttekintettük az egyszerű mintázatok létrehozásának és alkalmazásának alapjait, most már képesek vagyunk a véletlen segítségével modellezni, például egy aszfalt vagy a beton felületét, továbbá faerezetet hozhatunk létre vagy kedvenc szőnyegünk anyagát is elkészíthetjük.
A következő részben folytatjuk a mintázatok tárgyalását, megismerjük, hogyan lehet egy tetszőleges képet felületi mintázatként felhasználni, milyen módon tudunk a tükröződő felületen valódi tükröződést megjeleníteni, és miként lehet bővíteni a meglévő mintázatok tárházát.
Addig is kellemes kísérletezést és alkotást kívánok mindenkinek!

Fábián Zoltán (dzooli@freemail.hu, dzooli@yahoo.com) 27 éves, jelenleg programozóként dolgozik. Szabadidejében szívesen kirándul, túrázik. Emellett szeret rajzolni, érdekli a 3D grafika és a Linuxszal kapcsolatban minden olyan program és programnyelv, amit még nem ismer vagy nem próbált ki.

Memóriakezelés a gyakorlatban

Sorozatunk előző részében csak elméletben foglalkoztunk a virtuálismemória-kezeléssel. Ennek most vége, a következőkben bemutatjuk, miként is zajlik mindez a gyakorlatban.

Példánk alanyainak valódi éles operációs rendszereket választottunk, amelyekben olyan apróságokat is figyelembe kell venni, mint a hatékonyság és a biztonság. De mindenezek előtt beszéljünk egy fontos szempontról, amit mindig figyelemben kell tartani a memóriakezelő tervezésekor. Ez pedig az úgynevezett hordozhatóság. Az elmúlt hónapban hosszasan ecseteltük, hogy a mai processzorok már nagyon okosak, és sok mindenben megkönnyítik az operációs rendszer memóriakezelőjének életét. Példát is mutattunk, mégpedig az Intel Pentium processzorcsaládot, ami amellett, hogy támogatja a szakaszolt lapozást, még a védelem kérdését is megoldja – az úgynevezett védelmi szintek bevezetésével. Ezek után azt gondolhatnánk, hogy memóriakezelőt írni „könnyű". Az biztos, hogy nem gyerekjáték, de sokat segíthet a dolgon, ha a munka oroszlánrészét Pentium processzorunkkal végeztetjük el. De ha olyan memóriakezelőt írunk, ami teljes egészében a processzor szolgáltatásaira támaszkodik, jelentősen csökken rendszerünk hordozhatósága. Csakhogy nem minden processzor képes olyan magas szintű memóriakezelésre, mint a Pentium. Sőt akad olyan is, amelyik semmi ilyesmire sem képes. Például ilyenek a 80386-osnál régebbi Intel processzorok. Ha az egész memóriakezelésünket a Pentium szolgáltatásaira alapozzuk, akkor le kell mondanunk arról, hogy a rendszerünket valaha is ennél régebbi processzoron futtassuk. Az is igaz viszont, hogy manapság ritkán van szükség olyan operációs rendszerre, aminek ilyen múzeumba illő eszközön is futtathatónak kell lennie.

(Létezik azonban olyan operációs rendszer is, amit kifejezetten arra terveztek, hogy bármilyen IBM PC-megfelelő gépen fusson, ez a Minix. Ahhoz, hogy ez lehetséges legyen, a Minix nem használ sem szakaszolást (segmentation), sem pedig lapozást. Sőt még tárcserét (swapping) sem, azaz a folyamatokat egyáltalán nem cserélgeti a lemez és a központi tár között. Még a memóriában sem mozgatja a betöltött folyamatokat, mindegyik ott marad, ahová indításakor került. Emiatt a Minix alatt sajnos nem futtathatók olyan alkalmazások, amiknek a memóriaigénye meghaladja a fizikai tár méretét. Ez egy „éles" rendszeren valóban komoly hiányosság lenne, de a Minix esetében, amit kifejezetten (oktatási célokra) kis személyi számítógépekre terveztek, nem az. Cserébe, hogy a Minix nem használ virtuális memóriakezelést, minden IBM PC-n futtatható, és könnyen átültethető más kisgépekre).

Nemcsak a régi, de az újabb processzorok virtuális memóriakezelésében is komoly különbségek vannak. Vegyük például az UltraSPARC processzorokat, amelyek 64-bites címtartománnyal bírnak. Míg a Pentium háromféle virtuális memóriakezelést is támogat (lapozás, szakaszolás, lapozott szakaszolás), addig az UltraSPARC-on kizárólag a lapozást használhatjuk.

A félreértések elkerülése végett: ez nem azt jelenti, hogy az UltraSPARC „elmaradottabb" a Pentiumhoz képest. Inkább arról van szó, hogy a Pentium 32-bites szakaszokkal (segment) dolgozik, ezért a méretük viszonylag kicsi (legfeljebb egymillió lap). Egy 64-bites címtartomány esetén a szakaszok több billió lap méretűre is duzzadhatnak, amiket nem lehet a hagyományos laptáblás módszerrel nyilvántartani. Egyébként a Pentium processzoroknál is előfordulhatna ehhez hasonló gond, ha a programoknak egyszerre több ezer szakasszal nyílna lehetőségük bűvészkedni. Szerencsére azonban sem a Windows, sem a Unixok nem engedik meg, hogy egy folyamat egy szakasznál többet birtokoljon.

A két processzor között különbség van a TLB-használatban is, pontosabban a TLB-hibák kezelésében. Míg Pentium esetében ilyenkor a TLB újrafeltöltése gépszinten megy végbe, addig a SPARC egyből az operációs rendszerhez fordul. Az észbe vésendő tanulság tehát az, hogy minél jobban kihasználjuk az adott processzor virtuális memóriakezelő szolgáltatásait, annál nehezebb lesz rendszerünket más kiépítésekre átültetni. Megfordítva: minél kevesebb dolgot tételezünk fel az adott gépről, annál jobban nő rendszerünk hordozhatósága.

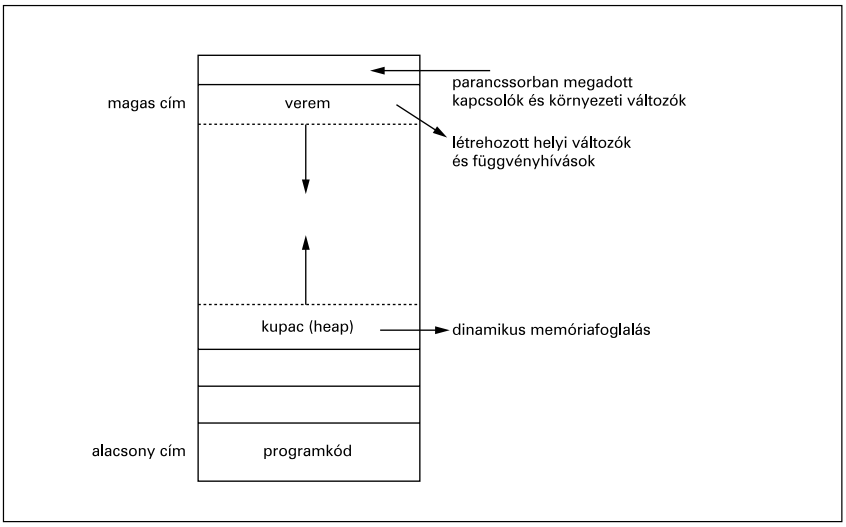
Virtuálismemória-kezelés Windows NT/2000/XP-ben

A Windows nem éppen korunk legnépszerűbb operációs rendszere, sőt bizonyos körökben egyenesen utálatnak örvend. Mi azonban mégis vagyunk olyan bátrak, hogy szóljunk néhány szót a Windowson belüli virtuális memóriakezelésről.

Lehet, hogy egy kicsit furán veszi ki magát, amiért egy, a „Linux-barátokat" megcélzó lapban Windowszal terheljük olvasóinkat. Ennek azonban részünkről megvan az alapos oka, mégpedig az, hogy az NT memóriakezelése nem rossz (most magáról az elvről beszélünk, a megvalósításról inkább nem nyilatkozunk), ezért hasznosnak találjuk – még mielőtt rátérnénk kedvenc operációs rendszerünk memóriakezelőjének boncolgatására –, hogy bemutassuk, miként is működik ez egy másik rendszerben. Fel szeretnénk hívni rá a figyelmet, hogy mi kizárólag az NT „családba" tartozó rendszerekről beszélünk, és nem a 3.1/95/98/ME nevű találmányokról.

A 3.1 igazából külön kategória, mivel gyakorlatilag csak egy MS-DOS-ra épített grafikus felület volt. A 95-től kezdve már olyan szolgáltatások is megjelentek, mint a virtuálismemória- vagy a folyamatkezelés. Az MS-DOS azonban továbbra is ott lapult legalul, tehát ha úgy vesszük, ez egyfajta kiegészítés volt, ami az MS-DOS-ból „valódi" operációs rendszert formált. Tette ezt azonban nagyon szerencsétlenül, ugyanis a 95-ös kódja 32-bites volt, ám az együttműködés megőrzése végett egy csomó 16-bitest is meghagytak, továbbá a legtöbb szolgáltatást magával az MS-DOS-sal végeztették el, például a fájlkezelést. A Windows 98 megjelenésével annyit

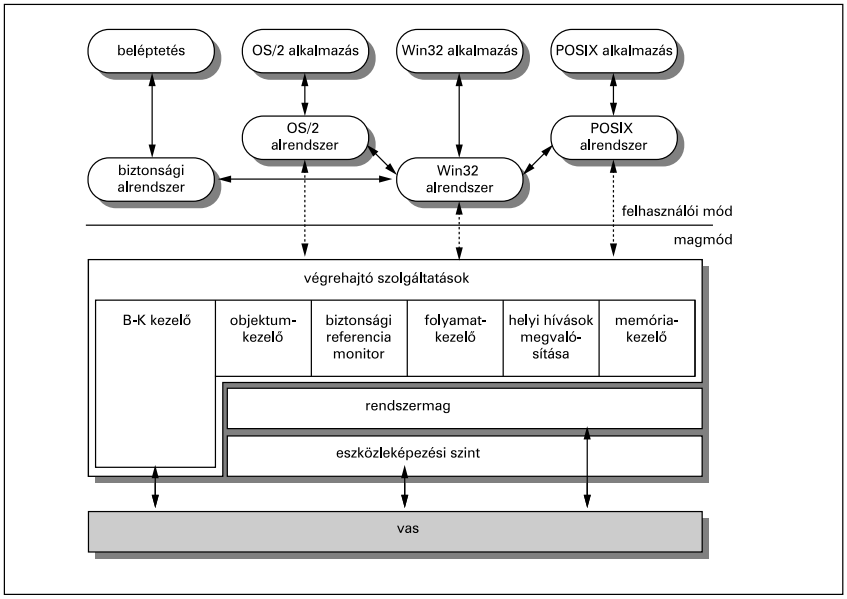
javult a helyzet, hogy bizonyos szolgáltatások magasabb szintre, a 32-bites kódok közé kerültek, így a rendszer üzembiztosabbá vált, de az MS-DOS-tól továbbra sem lehetett megszabadulni.
A Windows NT teljesen más, az égvilágon semmi köze sincs az MS-DOS-hoz. Saját fájlrendszerrel, memóriakezeléssel stb. rendelkezik, tehát valódi, önálló, 32-bites operációs rendszer.
Az NT kezdetben folyamatstrukturált felépítésű volt. Emlékezzünk vissza, ez azt jelenti, hogy a különböző alrendszerek önálló folyamatként futnak. Ha egy felhasználói folyamatnak szüksége van az egyik alrendszer szolgálataira, akkor üzenetet küld neki, majd a feladat elvégeztével visszakapja az eredményt, szintén üzenet formájában (ez az úgynevezett kiszolgálóalapú modell). Ennek az elgondolásnak az az előnye, hogy a rendszert rendkívül könnyen át lehet ültetni más felületekre.
A folyamatstrukturáltságnak azonban megvan az ára, mégpedig a hatékonyságbeli visszaesés. Ezért az NT 4.0-tól úgy határoztak, hogy amit lehet, beviszik a rendszermagba, ezáltal is sebességnövekedést valósítva meg. Arról lehet vitatkozni, hogy ez mennyire számított jó döntésnek, mindenesetre a 2000 és az XP megjelenésével a Microsoft a tömbszerű (monolitikus) memóriakezelés mellett kötelezte el magát.
Mint már említettük, az NT 32-bites operációs rendszer, tehát 32-bites virtuális címtartományokkal dolgozik. Minden folyamat saját címtartományt kap, ami annyit tesz, hogy egy folyamat legfeljebb 4 GB memóriával gazdálkodhat. Valójában a program kódjának és az adatoknak csupán 2 GB áll rendelkezésükre, a felső 2 GB a rendszermag memóriájához való hozzáféréshez kell (ez csak szigorú megkötések mellett lehetséges).
Az NT a memóriát 4 KB-os laponként kezeli. A lapozáson kívül azonban a memóriakezelő sokkal több szolgáltatást képes kezelni, például az úgynevezett íráskori másolás (copy on write) módszert, vagy a fájlok memóriaként történő elérését (lásd később). Ezeket a szolgáltatásokat a felhasználói folyamatok a Win32 API-n keresztül érhetik el.
Az első és legfontosabb feladat, amit minden memóriakezelőnek végre kell tudnia hajtani, a memóriafoglalás (memory allocation). Ez a Windows esetében két lépésben megy végbe. Az első az úgynevezett foglalás (reserve), amikor magát a virtuális címteret foglaljuk le. A második az egyeztetés (commit), amikor pedig maga a virtuális memória kerül lefoglalásra.
Felmerül két kérdés: mi az érdemi különbség a két lépés között, illetve mi az értelme ennek az egésznek? Az elsőre válaszolva: a foglalás csak azt mondja meg az operációs rendszernek, hogy várhatóan mennyi memóriára lesz szükségünk, de tényleges memóriafoglalás nem megy végbe. Maga a tárterület csak az egyeztetés folyamán lesz lefoglalva. Értelemszerűen egyeztetést már csak előzőleg lefoglalt területen lehet végrehajtani, de amíg az adott területet nem egyeztettük a memóriakezelővel, addig minden hivatkozás hibát fog okozni.



1. ábra
A folyamat rendelkezésére álló memóriarészt három (pontosabban négy) részre osztjuk fel. Legalulra kerül a programkód, utána jönnek az adatok és legfelülre a verem. Az adat és a verem egymással ellentétes irányban nő, a folyamat indításakor átadott értékek, illetve a környezeti változók pedig a verem fölé kerülnek

Vajon mit nyerünk ezzel? Hatékonyabb működést! A folyamatok futtatásának fizikai memóriaigénye ugyanis csökken, mivel tényleges erőforrás-használatra csak az egyeztetés után kerül sor. Nem is beszélve arról, hogy a rendszer terhelése nélkül foglalhatunk le nagy terjedelmű és egybefüggő címtartományokat.
Ha netán kétséges lenne a kétlépcsős memóriafoglalás előnyös mivolta, gondoljunk a szálak veremkezelési kérdésére. A vezérlési szálakról (vagy más néven a pehelysúlyú folyamatokról) már szó esett valamikor nagyon régen (Linuxvilág 2002. szeptemberi szám). Akkor azt mondtuk, hogy a szálak utasítások sorozatából álló valamik, amiket a gép majd szépen sorban végrehajt. Azt is mondtuk, hogy minden folyamatban legalább egy ilyen szálnak kell lennie, de egyes rendszereken több is lehet.
Ezáltal megvalósítható, hogy egy folyamat „egyszerre több mindent csinálhasson". A szálak hasznosságára a legjobb példa a grafikus webböngésző és a sok kis képet tartalmazó weboldal esete. Ha ez az oldalt meg akarjuk nyitni a böngészőnkkel, és a böngésző minden képet egymás után töltene le, akkor lehet akármilyen széles sávú elérésünk, a weben való böngészés nem lesz az örömök forrása. Ha azonban minden kép letöltését külön szál segítségével oldjuk meg, akkor amíg a kiszolgálótól visszaérkező válaszra várunk, bőségesen jut idő a többi kép letöltésére is.
A nehézség annyi, hogy minden szálnak saját veremmel kell rendelkeznie, aminek egy folyamatos címtartományon kell lennie. Az NT általában 1 MB-nyi vermet foglal minden szálnak. Ha a foglalás azonban egy lépésben menne végbe, akkor minden létrehozott szál elvisz 1 MB-ot a fizikai memóriából, miközben valószínű, hogy ennek az egytizedét sem fogja kihasználni. Például egy olyan weboldal letöltéséhez, ami tíz 4 KB-os képet tartalmaz, csak a szálak verméhez 10 MB-nyi memóriára lenne szükségünk.
Ezzel szemben a kétlépcsős megoldás esetén az 1 MB-ot csak lefoglaljuk (de nem egyeztetjük), egyeztetni csak kétlapnyi memóriát fogunk. A verem teteje az első lap lesz, a második azért szükséges, hogy tudjuk, mikor éri el a verem azt a méretet, amikor további lapok egyeztetésére is szükség lesz.

Ennek köszönhetően az előző példában említett szálak fizikailag csak 8 KB-ot fognak felfalni.
Érdemes szólnunk pár szót az NT lapkezeléséről is. Minden lapnak három állapota lehet: szabad, foglalt és egyeztetett. Ha egy lap szabad, akkor semmilyen adat sincs rá leképezve, és egy ilyen lapra való hivatkozáskor mindig hiba történik. A foglalt lapokra nem képezhető adat, nem lehet rájuk hivatkozni, és ez az állapot egészen addig így marad, amíg a lekötöttség meg nem szűnik.
Az egyeztetett lapokra történő hivatkozások a vas segítségével képeződnek le. Ez a művelet csak akkor sikeres, ha az éppen a memóriában tartózkodik. Ha nincs benne, akkor laphiba történik, és az operációs rendszernek a lapot be kell töltenie a lemezről. Minden memóriában lévő egyeztetett lapnak létezik egy mása a lemezen, amelyet az adott lap árnyéklapjának nevezünk (ide kerülnek majd, ha kivesszük őket a memóriából). Az árnyéklapok úgynevezett lapozófájlok formájában vannak tárolva. Fontos, hogy a foglalt, illetve a szabad lapokhoz nem tartozik árnyéklap (így amikor rájuk hivatkozunk, a rendszer nem tud mit betölteni a lemezről, tehát hiba keletkezik – értsd: kék képernyő).
Az NT azt is támogatja, hogy akár egy állományt is leképezhessünk a virtuális címtartományra. Ezt a következőképpen képzelhetjük el: tegyük fel, van egy 10 bájtos állományunk, és azt képeztük le a 140–150 virtuális címtartományra. Ezután, ha a folyamat kiolvassa a 140-es memóriacímen található adatot, akkor a fájl első bájtját kapja vissza (írásnál ugyanígy). Magyarán anélkül tudunk írni, illetve olvasni a fájlból, hogy bármilyen B-K műveletet hajtanánk végre. Fontos megjegyezni, hogy gyakorlatilag semmi különbség nincs az egyeztetett lapok és egy állomány virtuális címtartománya között. Egyszerűen csak arról van szó, hogy az utóbbi esetben az árnyéklapok nem valamelyik lapozófájlban, hanem magában az állományban foglalnak helyet. (Az is igaz, hogy ilyenkor könnyen előfordulhat olyan eset, hogy a memóriában más van, mint a lemezen. Ez azonban az állomány frissítésével könnyen orvosolható. Ezt az NT a `flush` nevű függvény segítségével oldja meg).
Az NT lehetőséget ad az osztott memória használatára is. Ez azt jelenti, hogy ugyanazt az állományt két vagy több folyamat is leképezheti saját címtartományába, vagy ugyanazt az állományt különböző virtuális címekre. Ez a módszer gyors megoldást kínál a folyamatok közötti kapcsolattartásra, mivel az osztott lapokon az egyik folyamat által előidézett változtatásokat az összes többi folyamat azonnal látja.
Az NT által támogatott megosztás másik formája az íráskori másolás (copy on write). Ez azt jelenti, hogy az osztott lapokat bármelyik folyamat bármikor olvashatja. Ha azonban valaki írni szeretne, akkor laphiba lép fel, és a rendszer a folyamat számára egy másolatot hoz létre a módosítandó lapról. Az érdemi különbség a két módszer között az, hogy ebben az esetben az összes folyamat úgy érzékeli, mintha egyedül ő képezte volna le az adott fájlt.



2. ábra
A Win32 alrendszer felhasználói módban fut és vele tartanak közvetlen kapcsolatot a windowsos alkalmazások. A többi (Posix és OS2) alrendszer is a Win32-re épül, de az ezeket használó alkalmazásoknak nincs lehetőségük közvetlenül a Win32-höz fordulniuk

Rendszerhívások a Windows NT-ben

Érdemes pár szót szólnunk arról is, hogy a felhasználói folyamatok (vagyis az alkalmazások) miképpen érhetik el az operációs rendszer szolgáltatásait.
Az NT esetében az alkalmazások rendszerhívásokat nagyon ritkán hívnak közvetlenül. Ennek talán a legfőbb oka az, hogy a Microsoft módszeresen elhallgatja az NT rendszerhívásainak a listáját (pontosabban nem teszi közzé őket teljes egészükben, és a különböző változatok megjelenésével folyamatosan módosítgatja őket). Az alkalmazások ezért a környezeti alrendszerekhez (environmental subsystems) fordulnak segítségül, amelyek különböző függvények hívását teszik lehetővé.
A legfontosabb alrendszer a Win32, amelyet a Windows-alkalmazások hívnak, de NT alá létezik egy Posix és egy OS/2 alrendszer is a Unix, illetve az OS/2-ből átültetett programok számára. (Az igazat megvallva a Posix alrendszer az NT-ben fabatkát sem ér. Egyrészt nem önálló alrendszer, a Win32-re támaszkodik, de az ezt használó alkalmazások számára nem teszi az összes Win32 függvényt elérhetővé, magyarán szólva a Posix alrendszerre támaszkodó programok nem tudnak szabadon Win32-es függvényeket hívni). Emiatt már-már lehetetlen egy komolyabb unixosos alkalmazás NT alá történő átültetése. Valószínűleg azért születhetett meg mégis ez a Posix rendszer, mert az Egyesült Államokban az állami intézményekben csak olyan rendszereket lehet használni, amelyek tartalmazzák a Posix-megfelelőséget. (A Microsoft nyilván nem akart elesni ettől a piactól).
A rendszerhívások nyilvánossá tétele helyett inkább egy felhasználói kezelőfelületet (interface) hoztak létre, amit Win32 API-nak (Application Programming Interface) neveznek. Ez jól van dokumentálva, és hihetetlen mennyiségű függvényt tartalmaz. Ezek a függvények vagy meghívnak egy rendszerhívást, vagy a Win32 alrendszerhez fordulnak, de a legtöbb a felhasználói módban maradva hajtja végre a feladatát.
A Win32 API eszmeisége kicsit furcsa (azok számára, akik nem Windowson

nevelkedtek). Míg a legtöbb operációs rendszer (például a Unix) esetében arra törekednek, hogy minél kevesebb rendszerhívás legyen (pontosabban csak annyi, amennyi feltétlenül szükséges ahhoz, hogy a rendszer minden szolgáltatását kényelmesen elérhessük), addig a Windowsra a halmozás a jellemző. Ez alatt azt értjük, hogy arra törekednek, hogy minél több függvénnyel árasszthassák el a programozót, amik fedjék le úgymond az „élet minden területét", és ugyanazt a dolgot több különböző módon is el lehessen végezni. Például külön API-függvény létezik arra, hogy egy fájlt teljes egészében egy másik helyre másoljunk át.
(A Win32 API egyébként a 95/98/ME rendszerekben is megtalálható, igaz, egy kicsit kevesebb szolgáltatással. Például nem érhetők el a különböző biztonsági szolgáltatások, mivel ilyesmit ez a rendszer egyáltalán nem tartalmaz, továbbá bizonyos függvényeket másképpen kell paraméterezni).

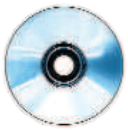
Virtuálismemória-kezelés a Unixban

Manapság rengeteg Unix van forgalomban, ezért elég nehéz általánosságban beszélni róluk. A következőt fogjuk tenni: kiválasztunk egyet közülük (legyen mondjuk a Linux), és annak a virtuális memóriakezelését fogjuk töviről-hegyire bemutatni. Ez lesz a következő rész témája.
Most azonban – bevezetésként – megnézzük, hogyan is kezeli a virtuális memóriát a Berkeley Unix.
Az első kérdés: mi az a Berkeley Unix? A történet az 1970-es évek elejére nyúlik vissza, amikor is egy Bell Labs nevű intézményben megszületett az első Unix, a labor tulajdonosa az AT&T volt. Ez a rendszer egy PDP-11 nevű gépen futott, amelyek az Egyesült Államok egyetemein nagy népszerűségnek örvendtek. Ezért az AT&T a Unixot forráskódostul olcsón az egyetemek rendelkezésére bocsátotta, azok pedig egyből kaptak az alkalmon (mivel a PDP-11 eredeti operációs rendszere hihetetlenül használhatatlan volt).
A hangsúlyt arra helyeznénk, hogy a Unixot a forráskóddal adták oda, tehát minden egyetem kedvére továbbfejlesztgethette. Így történt ez a kaliforniai egyetemen is (ez a Berkeley). Itt vezették be a lapozott virtuális memóriát, és ez volt az első rendszer, ami támogatta a TCP/IP-n keresztüli kapcsolattartást, amely később az egész Internet alapjául szolgált.
(Eközben az AT&T is tovább fejlesztette a Unixot, ebből lett a System V. Mindkét Unix-változatról elmondható volt, hogy egyik sem működött együtt a másikkal. A helyzet orvoslására létrehozták a Posix – Portable Operating System-IX – nevű szabványt, ami meghatározta a rendszerhívásokat és az alapvető segédprogramokat. A Posixban meghatározott rendszerhívásokon kívül számos Unix-változat más rendszerhívásokat is lehetővé tesz, de ami a Posixban van, az mindenhol megtalálható).
A Berkeley Unix a következő memóriamodellt használja: minden folyamathoz három szakasz tartozik. Az elsőben a kód, a másodikban az adat és a harmadikban a verem foglal helyet (ezek közül a kódszakasz mérete biztosan nem fog változni a folyamat futása során).
Ha az adott gép nem támogatja a szakaszolást (csak egy virtuális címtérrel rendelkezik), akkor a kódot berakják a memória aljára. A legtetejére a szakasz kerül, középre pedig az adat. Mindez úgy helyezkedik el, hogy az adat és a verem egymással ellentétes irányban növekedhessen.
Ha az a gép, amin a rendszer fut, képes a lapozásra, akkor nyugodtan lapozható az egész címtartomány, hiszen a folyamatok ebből semmit se fognak észrevenni. Ha nincs lehetőség a lapozásra, akkor a folyamatokat sajnos csak teljes egészükben cserélhetjük a lemez és a központi tár között. Ez az úgynevezett tárcsere (swapping), amiről már szó volt két résszel ezelőtt. Az effajta memóriakezelésnek az a legnagyobb előnye, hogy szinte minden kiépítésen könnyedén megvalósítható.
A jelenlegi Unixok azonban ennél többre is képesek. Például a Solaris (ami a System V-re épül) memóriakezelője szintén képes olyan dolgokra, mint az NT-é. Itt is lehetőség nyílik az állományok virtuális címtérbe való leképezésére vagy az íráskori másolás módszerének használatára.
Sorozatunkat a következő alkalommal a Linux memóriakezelésének részletes ismertetésével folytatjuk.

➲ http://www.eg3.com/adv-goo/posix.htm

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.

Postfix és Cyrus



Levélkézbesítő ügynök, IMAP- és POP3-kiszolgáló, valamint virtuális felhasználók. Ez volt a feladat Debian Woody alatt. Most egy sör mellett próbálom meg felidézni a nehezét.

Beteszem a kedvenc FatBoy-albumomat, hátradőlök a gurulós székben, és diadalittasan mosolygok a monitorra. A munka első megközelítésben nem volt bonyolult, viszont annyi apró részfeladatból állt, hogy úgy érzem, megérdemelten koccinthatom oda a sörösdobozt a monitor oldalához. Ha te is valami hasonló tett előtt állsz, de már szeretnél ott tartani, ahol én, ez a cikk neked szól.

SMTP és IMAP, POP3

Előttem már többen kellő mélységben tárgyalták az említett protokollokat, ezért nem szeretném őket megismételni. Ha időd engedi, feltétlenül olvasd el ***Deim Ágoston*** kapcsolódó cikkeit a ***Linuxvilág*** 2001. június–júliusi számától kezdve 2001 novemberéig. Részletesen beszél a Postfixről mint MTA-ról, és egy összehasonlító írás erejéig kitér a főbb IMAP-, illetve POP3-kiszolgálókra. Ez a cikk viszont a türelmetlen rendszergazdáknak íródott, ezért lássuk, mit is takarnak a fenti rövidítések.

Az elektronikus levelezés alapvetően két összetevőből áll. Az MTA (Mail Transfer Agent) feladata a levél kézbesítése a hálózat (legyen az helyi vagy maga az Internet) különböző részhálózatain keresztül. A MUA (Mail User Agent) pedig az a levelező-program, amit a felhasználó az elektronikus levél fogadására, elkészítésére, illetve postázására használ (pl.: Sylpheed, mutt). Az MTA-k közötti üzenetváltás protokollja az SMTP.

Ha minden egyes hálózatba kötött számítógépen lenne egy MTA, akkor itt véget is érne a történet. Ez azonban kivitelezhetetlen és felesleges. Kivitelezhetetlen, ha figyelembe vesszük, hogy az Internet használóinak nagy része még mindig nem rendelkezik állandó kapcsolattal és változó IP-címe van. Ugyanakkor felesleges azért, mert egy számítógépet többnyire nem használnak 2–3 embernél többen. Így egy MTA-nak csak 2–3 postaládát kellene kezelnie, ami erőforrás-pazarlás.

A megoldás meglehetősen egyszerű: egy központi gépet kinevezünk levélkiszolgálónak. Állandó kapcsolattal rendelkezik és DNS-rekordja is van, ami név alapján megmondja az IP-címét. Tárolja a beérkező leveleket és kézbesíti a kimenőket. A belső hálózatra kötött felhasználók ugyanúgy SMTP-t használva küldenek levelet, csak beállítják az MUA-ban, hogy melyik az SMTP-kiszolgáló. Beérkező leveleiket pedig egy külön e célra kifejlesztett protokollon át töltik le a saját gépükre. Ez utóbbi lehet IMAP vagy POP3.

Vajon IMAP-ot használjunk vagy POP3-at? Alapvető különbség a kettő között, hogy az IMAP-ot azzal az elképzeléssel fejlesztették ki, hogy a levelek mindig is a központi számítógépen maradnak, és a felhasználók távolról kezelik a leveleiket. Ezzel szemben a POP3 alapvetően arra szolgál, hogy a felhasználó az összes levelét a saját gépére tölti le, letöltés után pedig törli őket a kiszolgálóról. A válasz tehát ebben a megközelítésben a levelezőkiszolgálón lévő tárhely függvénye.

Válasszunk egy MTA-t!

Legyen mondjuk a Postfix. Miért? Mert megbízható, kellően moduláris, illetve gyorsan és egyszerűen beállítható. Nekem egy 166 MHz-es Pentiumra kellett felkönyörögnöm egy ezer felhasználót kiszolgáló rendszert. Határozottan úgy érzem, hogy a Postfix jó választásnak bizonyult, mert nem érkezett panasz a sebességet illetően.

Először telepítsd a `postfix`, `postfix-ldap`, `postfix-pcre` és a `postfix-doc` csomagokat. Akármi is volt eddig az MTA-d, mostantól a Postfix lesz az. A `postfix-ldap`-ot és `postfix-pcre`-t a Postfix igényli, ezért kötelező őket telepíteni, habár most nem fogjuk egyik előnyét sem kiaknázni. Az egyik segítségével LDAP-ból lehet venni bizonyos adattáblákat (például az alias táblát), míg a másikkal szabályos kifejezéseket (regular) lehet használni ugyanezekben a táblákban. Megjegyezném, hogy a `postfix-doc` a leírást nem a */usr/share/doc/postfix-doc*, hanem a */usr/share/doc/postfix* könyvtár alá telepíti.

A telepítés során a `General type of configuration?` kérdésre az `Internet Site` – első nekifutásra – jó választás, a finomhangolás így is, úgy is kézzel fog történni. Ezt meglepően egyszerű kérdések követik. Előbb a kiszolgáló nevét kérdezi meg, azután azt, hogyha valaki egy egyetlen tagból álló tartománynévre akar levelet küldeni, akkor hozzáragassza-e a saját tartománynevét. Így például a `pisti@mail` címet egészítse-e ki `pisti@mail.cegnev.hu`-ra. Ez akkor lehet érdekes, ha a belső hálózaton lévő felhasználóinknak nincs erejük beírni a teljes címet, amikor a szomszédnak küldenek elektronikus levelet. Ezt követően olyan tartománynéveket kell megadnod, amelyeket a Postfix a sajátjának tekint, vagyis ha valaki a @ után ezt írja be, és a levél eljut ehhez a kiszolgálóhoz, akkor már nem küldi tovább, hanem megtartja. Az alapértelmezés többnyire megfelelő. Az utolsó kérdés azokra az IP-címtartományokra vonatkozik, amelyek SMTP-kiszolgálónak használhatják a Postfixet. Ez jellemzően a 127.0.0.1/8 és a belső hálózat. Az IP-címet egy / (perjel) után követő szám határozza meg, hogy a címből hány bit az alhálózati maszk. Ez például 10.x.x.x formájú címeknél nyolc, de 192.168.x.x esetén tizenhat.

A `debconf` sokat segített, de azért maradt még feladat bőven neked is. A */etc/postfix/main.cf* állományt kell módosítanod, ez a Postfix fő beállítófájlja. Mivel a Cyrust később arra is használni szeretnénk, hogy a helyi felhasználók leveleit a megfelelő postafiókban tárolja, a következő változtatások szükségesek:

```
#mailbox_command = procmail -a "$EXTENSION"
mailbox_transport = cyrus
```

Vagyis megjegyzésbe kell tenni a `mailbox_command`-ot, és helyette fel kell venni a `mailbox_transport`-ot. Ezzel a Postfix kész is, de a java még hátravan, hiszen most jön a Cyrus.

Válasszunk IMAP- és POP3-kiszolgálót!

Legyen mondjuk Cyrus. Miért? Mert megbízható, többféle hitelesítési módszert támogat, és az ACL-ek (Access Control List) segítségével finoman szabályozhatók a jogosultságok. Az említett Pentium 166-oson is a Cyrusra esett a választásom. Telepítened kell a `cyrus-common`, `cyrus-admin`, `tcl8.0` és a `tclreadline` csomagokat, a `cyrus-imapd`-t, illetve a `cyrus-pop3d`-t – attól függően, hogy melyik protokollt szeretnéd elérhetővé tenni. A Tcl-csomagok a `cyrus-admin` végett szükségesek, ugyanis a `cyradm` nevű felügyeleti felület Tcl-t használ. Ezek mind kérdezés nélkül települnek, úgyhogy kézzel kell egy kicsit varázsolni.

A `cyrus-imapd`, illetve a `cyrus-pop3d` a Cyrus IMAP-et és POP3-at megvalósító moduljai. Mind a kettő `inetd`-ből indul. A telepítés során a */etc/inetd.conf* a következő sorokkal egészül ki:

```
pop3    stream tcp    nowait cyrus
➥/usr/sbin/tcpd      /usr/sbin/pop3d
imap2   stream tcp    nowait cyrus
➥/usr/sbin/tcpd      /usr/sbin/imapd
```

A két modulnak közös beállítóállománya a */etc/imapd.conf*. Itt a következő módosítások szükségesek:

```
admins: bela
popminpoll: 0
```

Az `admins` mező alapértelmezetten megjegyzésben van, feltétlenül ki kell tölteni, különben nincs olyan felhasználó, aki felügyelhetné a Cyrust a `cyradm`-on keresztül. A 0-s azonosítójú felhasználót (root) biztonsági szempontból itt nem ajánlott megadni. A Cyrus leírásában azt is megemlítik, hogy a rendszerfelügyelőnek ne hozzunk létre postafiókot. A legegyszerűbb egy rendszerben is létező, semmilyen egyéb joggal nem rendelkező felhasználót `admin`-nak felvenni. A `popminpoll` azt az időt határozza meg percben, amelynek egy felhasználó esetén két bejelentkezés között el kell telnie, ha a POP3-protokollt használja. Az alapértelmezés egy perc, ami éles rendszer esetén jó a nyers erő- (brute-force) támadások visszaszorítása érdekében, de amíg építed a kiszolgálót, és azt próbálgatod, hogy mely részei működnek, melyek nem, addig nem biztos, hogy ez lesz álmaid netovábbja. Én erre az időszakra ezt a korlátozást kikapcsoltam.

A terjesztésben is megtalálható, a ***SASL*** nevű függvénykönyvtárat nélkülöző Cyrus-változat kétféle hitelesítést támogat: a Kerberos-, illetve a Unix-típusút. A Unix-típusú azonosítás a `pwcheck` démonnal működik. Vagyis nem elég az `inetd`-t futtatni ahhoz, hogy a Cyrus üzemeljen. A */etc/init.d/pwcheck* néven található a vezérlő parancsfájl. A `pwcheck` démon két változatban található meg a Woodyban. A „standard” változat egy megadott ***passwd***, illetve ***shadow*** állománnyal dolgozik.

A PAM-os változattal viszont tetszőleges PAM-modult alkalmazhatunk a hitelesítéshez. Ha belenézel a */etc/init.d/pwcheck*-be, azt láthatod, hogy a futtatandó program neve */usr/sbin/pwcheck*. Ez azonban egy közvetett hivatkozás a */etc/alternatives/pwcheck*-re, ami pedig a */usr/sbin/pwcheck_standard*-re mutat. Ezért, ha a PAM-os változattal szeretnél dolgozni (én ezt ajánlom), akkor a következőt tedd:

```
# rm /etc/alternatives/pwcheck
# ln -s /usr/sbin/pwcheck_pam
➥/etc/alternatives/pwcheck
# /etc/init.d/pwcheck restart
```

A Cyrus és a PAM

Már csak egy feladatod maradt: meg kell oldanod a felhasználókezelést. Mivel a `pwcheck` most már PAM-ot használ, csupán keresni kell egy olyan PAM-modult, ami az `auth` részben használható és a */etc/passwd*-től eltérő adatbázist használ. Én először a `pam_userdb` modult próbáltam ki, ami a `libpam-modules` csomag része, vagyis alapból feltelepül. Ez egy tetszőleges Berkeley hash DB állományt képes feldolgozni, ahol a felhasználónevek a kulcsok és a jelszavak az értékek. Ezzel csak az a baj, hogy egyszerű szöveges formában tárolja a jelszavakat, vagyis ha valakinek sikerül megszereznie az állományt, az összes jelszó a birtokába jut. Tovább keresgélve ráakadtam a `pam_pwdfile.so`-ra, amellyel egy */etc/passwd*-hez hasonló felépítésű állományt használhatunk. Az utóbbi modul a `libpam-pwdfile` csomag része. Tulajdonképpen egyetlen fájlt tartalmaz, a */lib/security/libpam_pwdfile.so*-t, de ez elég is. A DES és az MD5 titkosítást is támogatja. Használatához mindössze a */etc/pam.d/cyrus* állományt kell szerkesztened, a következőképpen:

```
#auth  required      pam_unix.so nullok
auth   required      pam_pwdfile.so pwdfile
➥/etc/imapd.passwd flock
#account      required      pam_unix.so
account       required      pam_permit.so
```

Sok PAM-hívő most legszívesebben a pokolba küldene a `pam_permit.so` használatáért, de az `account` részben véleményem szerint nincs szükség semmiféle ellenőrzésre. Visszatérve a `pam_pwdfile-ra`: ennek összesen két kapcsolója van. A `pwdfile`-t szóköz után követi a használandó állomány. A `flock` nem kötelező kapcsoló. Engedélyezi a `flock()` függvény használatát, vagyis bekapcsol egy olyan szerkezetet, amivel megbizonyosodhatsz róla, hogy a megadott adatbázishoz egyidejűleg nem fordul két különböző folyamat (lásd `man 2 flock`). A kapcsolóval ellenkező hatást fejt ki a `noflock`. Ahogy a súgóoldalon is olvashatod, ez a biztosíték csak addig él, amíg az összes folyamat használja a `flock`-ot. Ezért a */etc/imapd.passwd*-vel nagyon óvatosan kell bánni.

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

use strict;
use Fcntl ':flock'; # A LOCK_* Ælland khoz

die "HasznÆlat: ".$0." {felhasznÆl }
➥{jelsz }\n" if $#ARGV != 1;

open (PWD,">>/etc/imapd.passwd");
flock(PWD,LOCK_EX); # csak Øn hasznÆlhatom
                    ➥az ÆllomÆnyt
seek (PWD,0,2); # ha valaki k zben elvitte
                ➥volna a mutat t
my $revord = $ARGV[0].":";
my $salt = join '',
➥('.','/',0..9,'A'..'Z','a'..'z')[rand 64,
➥rand 64]; # s  lØtrehozÆsa
$record .= crypt ($ARGV[1],$salt)."[13]\n";
print PWD $record;
flock (PWD,LOCK_UN); # lock kikapcsolÆsa
close (PWD);
```

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

use strict;
use IMAP::Admin;

die "HasznÆlat: ".$0." {felhasznÆl }\n"
➥if $#ARGV;

my $mb = "user.".$ARGV[0];
my $imap = IMAP::Admin->new('Server' =>
➥'localhost',  'Login' => 'bela',
                'Password' => 'almasretes');

die $imap->{'Error'} if $imap->create($mb);
die $imap->{'Error'}
➥if $imap->set_quota($mb,2000);

$imap->close;
```

Ahogy a */etc/passwd*-t csak `vipw`-vel szabad szerkeszteni, úgy ezzel is kellő körültekintéssel kell bánni. Az egyetlen bökkenő, hogy az állomány formátuma korántsem megszokott, és a *passwd*-re is csak messziről hasonlít:

```
felhasznÆl nØv : titkos tott_jelsz  [ szÆm ]
```

A szögletes zárójelek közé zárt szám jelzi, hogy milyen algoritmust használtak a jelszó titkosításakor. Akkor 13, ha a Unix `crypt()` függvényével történt, és 34, ha MD5-tel. Hogyan hozzunk létre bejegyzéseket az állományban, ügyelve a `flock()` használatára, és minél kevesebb idő alatt? A válasz a Perl (*1. lista*). Azért van szükség a `flock()` utáni `seek()`-re, mert a `flock()` függvény egészen addig nem tér vissza, amíg meg nem szerzi az „exkluzív zárat”. Ha azért nem kapja meg azonnal, mert valaki más már rátette a saját zárát az állományra, akkor megvárja, amíg leveszi róla. Ha a másik folyamat közben a fájlmutatót elvitte az állomány végéről (hozzáírásra nyitottuk meg), akkor vissza kell tenni. A `crypt()` függvényt használom titkosításra. Ez két értéket vár: a titkosítatlan szöveget és egy kis sót (salt). A só egy két karakterből álló véletlen szöveg.
A felhasználó-adatbázis most már feltölthető felhasználókkal. Mindenekelőtt vedd fel a felügyelődet, mert a `cyrus` most már nem a */etc/passwd*-t használja, így jelenleg egyetlen felhasználót sem ismer meg. Ha megvan az `admin`, létrehozhatsz mezei felhasználókat is, de ezeknek még nem lesz postafiókjuk. Ugyanis nemcsak a felhasználókat kell létrehozni, de a postafiókot is. A postafiók létrehozása a `cyradm` nevű programmal tehető meg:

```
# cyradm localhost
localhost userid: bela
localhost password: almasretes
localhost>
```

Természetesen az `almasretes` nem jelenik meg a képernyőn. Itt a legegyszerűbb meghívni a súgót, ez megadja az összes használható parancsot. A `pisti` felhasználónévhez tartozó postafiókot a `cm user.pisti` paranccsal lehet létrehozni (a `cm` a `create mailbox` rövidítése).
Most már nagyon közel vagy a sörhöz. Csakhogy a főnököd most adott a kezedbe egy hajlékonylemezt. A lemezen található állomány a felhasználóneveket és a jelszavakat tartalmazza, úgy ezret. A `cyradm` ehhez túlságosan is felhasználói beavatkozást igénylő (interactive). A válasz a Perl. Most egy olyan parancsállományt hozunk létre, ami csak egy felhasználónevet vesz át értékként, és létrehozza a megfelelő postafiókot, sőt még egy két megabájtos tárkorlátot is beállít. Jól hangzik, már csak a megfelelő Perl modul hiányzik. Telepítsd a `libimap-admin-perl` csomagot (*2 lista*).
A program magáért beszél, a `$imap objektum {'Error'}*` tulajdonsága a legutolsó meghívott elemfüggvény sikerességétől függően tartalmaz egy szöveges hibaüzenetet.

Jöhet a sör!

Az utóbbi két Perl-parancsfájlt használva lehetőségeid már korlátlanok. Bátran írj, ha valami nem úgy működik, ahogy leírtam, vagy valamit nem sikerült jól elmagyaráznom.
Jó sörözést!

A Cyrus és a Postfix programok forráskódjai a 47. CD Magazin/Cyrus könyvtárában találhatóak.

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

Héjprogramozás Linux alatt

A dolgok értelme, avagy miért érdemes megtanulnunk a héjprogramozást?

Egy operációs rendszer két legalapvetőbb feladata a számítógép erőforrásainak elosztása és a felhasználóval való kapcsolatteremtés. Bármilyen operációs rendszerrel legyen is dolgunk, biztosan lesz egy olyan része, amelyik a géppel, és egy olyan, amelyik a felhasználóval képes szót érteni. Ezek természetesen egymással is beszélő viszonyban vannak, másként a rendszer nem működhetne, de ez a belső viszony a gép előtt ülő számára mindvégig rejtve marad.
A Linux esetében az eszközökkel a rendszermag (kernel), a felhasználóval a héj (parancsértelmező) tartja a kapcsolatot. A héj egyfajta tolmácsként működik: értelmezi a felhasználó által begépelt parancsokat, ellenőrzi az írásmódjukat, majd ha mindent rendben talált, elindítja a szükséges programokat.

Parancsoljunk!

Mindebben semmi érdekes nincs egészen addig, amíg csak egy-egy parancsot akarunk végrehajtani. Begépeljük például hogy `ls`, és a Linux „rögtön tudja”, hogy a pillanatnyi könyvtár tartalmát szeretnénk a képernyőn viszontlátni. Az egész valahogy úgy működik, mint egy italautomata: bedobod a pénzt, megnyomod a gombot, és – egy kis szerencsével – megkapod, amire vágytál. Némi túlzással azt mondhatjuk, hogy a Unixtól különböző operációs rendszerek alapvetően valóban így működnek. Tényleges gondokra kézzelfogható megoldásokkal szolgálnak előre megírt programok formájában. Ezek természetesen kiválóan használhatók egészen addig, amíg a felhasználónak „ötletei nem támadnak”. Ha ugyanis olyasvalamit akar megvalósítani, ami egy-egy ilyen készen kapott eszközzel már nem megy, akkor vagy megkísérli ügyesen összekombinálni azokat, vagy ha ez nem sikerül, egyszerűen megírja magának a megfelelő programot, például C-ben.
A Unix rendszerek, így a Linux fő erőssége a dolgok kombinálása. Az operációs rendszerrel kapunk egy nagy halom, jobbára teljesen „szakbarbár” segédprogramot. Némelyik betűket cserélget a bemenő szövegben mindenféle körmönfont szabályok alapján, de számolni gyakorlatilag nem tud (`sed`, `tr`). Mások a számokat is ismerik (`expr`), megint mások tulajdonképpen önálló programozási nyelvnek is tekinthetők (`awk`).
Különlegességük folytán ezekkel az eszközökkel önmagukban semmire sem mennénk. Szerencsére van azonban egy olyan rendszerszolgáltatás, a cső (jele a | karakter), ami lehetővé teszi az egyes elemek sorba kapcsolását. Ennek segítségével az egyik program kimenetét rögtön egy másik bemeneteként használjuk, anélkül, hogy az fizikai formában (például átmeneti fájlként) bárhol megjelenne. Ha például a pillanatnyi könyvtárban található bejegyzések közül ki akarjuk válogatni az alkönyvtárak neveit, így járhatunk el:

```
ls -l | grep ^d
```

Már ez a végtelenül egyszerű példa is jól szemlélteti, hogy a „csövezés” segítségével az önmagukban buta elemekből nagyon is okos rendszereket építhetünk anélkül, hogy olyan alacsony szintű műveletekkel kellene foglalkoznunk, mint a fájlok megnyitása, vagy a belőlük való olvasás.
És ezzel el is érkeztünk most induló cikksorozatunk témájához, a héjprogramozáshoz, ami tulajdonképpen nem egyéb, mint az előbb említett „csövezés” egy kissé hivatalosabb megnevezése.

Írjunk egy egyszerű héjprogramot!

A héjprogramok első közelítésben egyszerű szövegfájlok, amelyek a héj számára értelmes utasítássorokat tartalmaznak. Ezeket akár a parancssorba is begépelhetnénk, de ha hosszúak vagy sokszor kell az adott műveletsort használnunk, célszerűbb belőlük héjprogramot írni. A fenti egyszerű példánál maradva a könyvtárválogató trükköt a következőképpen alakíthatjuk állandó segédeszközzé (a sorokat csak a jobb megértés miatt számoztuk):

```
1: #!/bin/sh
2: # Az alk nyvtÆrak neveinek listÆzÆsa
3: ls -l | grep ^d
```

Az első sor annak a parancsértelmezőnek a nevét és teljes elérési útvonalát adja meg, amelyik képes értelmezni a leírtakat. Linux esetében ez csaknem mindig a Bash, a */bin/sh* itt ugyanis valójában egy erre a héjra mutató hivatkozás. A második sor megjegyzés, amit a kettős kereszt (# karakter) után írhatunk. Ennek nem muszáj a sor elején kezdődnie, de a szöveg csak úgy lehet több soros, ha mindegyik elejére kiírjuk a kettős keresztet. A lényeget a harmadik sor tartalmazza. Hozzunk tehát létre egy tetszőleges szövegszerkesztővel egy ilyen tartalmú szövegfájlt. Legyen a neve, mondjuk *konyvtar.sh*. Ahhoz, hogy futtatni is tudjuk, végrehajtási jogot kell neki adnunk:

```
chmod 700 konyvtar.sh
```

Ezzel el is készült első héjprogramunk. Próbáljuk ki! Ahhoz, hogy bárhonnan elérhető legyen, természetesen egy olyan könyvtárban kell elhelyeznünk, amelyik szerepel a `PATH` környezeti változóban.
Mindez természetesen még csak a kezdet. A héjprogramok fejlesztése során számos, más programozási nyelvben is megtalálható vezérlési szerkezetet lehet és kell használnunk. Létrehozhatunk ciklusokat, döntési szerkezeteket, átmeneti fájlokat, kezelhetjük a parancssort és társaloghatunk a felhasználóval. Cikksorozatunkban az alapok tisztázása után az olyan elvontabb, de általánosan használható módszerek kerülnek sorra, mint a zárolás, a parancssori kapcsolók kezelése, vagy az önhívó programok írása.

Büki András (buki@vuk.chem.elte.hu)
Körülbelül kilenc éve dolgozik Linux operációs rendszerrel. Állandó szerzőtársa Prof. Dr. H. V. Kuksinak, akivel a Duna vagy a Tisza partján szoktak az élet és a tudomány viselt dolgairól töprengeni.

A Sambáról röptében

Ha az irodánkban valami oknál fogva a linuxos (és más unixos) gépek mellett be kell fogadnunk néhány Windowst is, és meg kell osztanunk velük az erőforrásainkat, akkor a Samba jó szolgálatokat tehet nekünk.

A Samba segítségével könnyedén megoszthatjuk fájlrendszerünket és nyomtatóinkat. A Samba egy úgynevezett SMB (Server Message Block) nevű protokollra épül, ami a NetBIOS csatolófelületen keresztül teszi lehetővé, hogy egyenrangú (peer-to-peer) hálózatokban megoszthassuk a fájlrendszerünket, illetve a nyomtatóinkat. (Az egyenrangú hálózat azt jelenti, hogy nincs egy fizikailag kitüntetett kiszolgáló, minden gép szabadon elérhetővé teheti az erőforrásait a többiek számára).

Pár szó a NetBIOS-ról

Mi is az a NetBIOS, amelyre az SMB is épül? A Network Basic Input/Output System (hálózati alapvető ki- és beviteli rendszer) az IBM találmánya, ami hajdanán a PC-khez gyártott hálózati csatolókon lévő ROM-ban volt jelen. Ez tulajdonképpen egy olyan felület, ami lehetővé teszi a programozók számára, hogy gyorsan és kényelmesen valósítsák meg a hálózati kapcsolattartást. (A NetBIOS annyira magas szintű, hogy az OSI modell mind a hét rétegét lefedi).

Ma már nagyon ritka a „színtiszta" NetBIOS-t használó hálózat, ám emulálás segítségével a NetBIOS-on keresztüli kapcsolattartás szinte bármilyen hálózaton megvalósítható. Legyen az akár TCP/IP-s, IPX-es, akár DECNet-es. A NetBIOS emulálásának azonban hátrányai is vannak, mivel egyrészt meglehetősen lassú, másrészt nem tudjuk megoldani a több hálózati szakaszon keresztüli kapcsolattartást (emiatt az Internet és az helyi hálózatok térhódítása után a NetBIOS jelentősen visszaszorult). Mindezek ellenére a NetBIOS-t a legtöbb operációs rendszer támogatja, beleértve a Windowst és a Linuxot is, így jó alap az SMB protokoll számára.

A NetBIOS a gépeket egy 15 karakter hosszúságú egyedi névvel azonosítja. A gépeket munkacsoportokba (workgroups) szervezhetjük. Ezeket a munkacsoportokat is névvel azonosítjuk. A munkacsoportoknak az a lényegük, hogyha nem egy meghatározott gépnek, hanem egy munkacsoportnak küldünk egy üzenetet, ezt az adott munkacsoportba tartozó összes gép meg fogja kapni.

Az SMB

Az SMB egy olyan protokoll, amelyet eredetileg a Windows és az OS/2 használt fájlok és nyomtatók megosztására. A legtöbb operációs rendszer ezt is támogatja valamilyen formában (vagy alapból, vagy valamilyen kiegészítő program segítségével).

Az SMB protokoll lehetővé tesz némi védelmet is megosztott erőforrásaink számára. Kétféle biztonsági szint létezik: az első a megosztási szint (share level), itt az adott könyvtárhoz vagy nyomtatóhoz egyszerűen csak hozzárendelünk egy jelszót. E jelszó ismerete nélkül a hálózaton keresztül senki sem férhet hozzá az adott erőforráshoz.

A másik szint egy kicsit nagyobb biztonságot jelent, mivel csak az férhet hozzá az eszközökhöz, akinek érvényes azonosítója van a linuxos gépen. Ez a felhasználói szint (user level), ahol először is mindenkinek azonosítania kell magát egy név és a hozzá tartozó jelszó segítségével.

A protokoll azt is lehetővé teszi, hogy az ügyfelek megtudhassák, illetve a kiszolgálók hirdethessék magukról, hogy milyen erőforrásokat osztanak meg (tallózás). Ezt úgynevezett üzenetszórással (broadcasting) valósítják meg. (A TCP/IP alapú hálózatokon ez néha gondot jelenthet, mivel az útvonalválasztók nem mindig engedik át ezeket az üzeneteket).

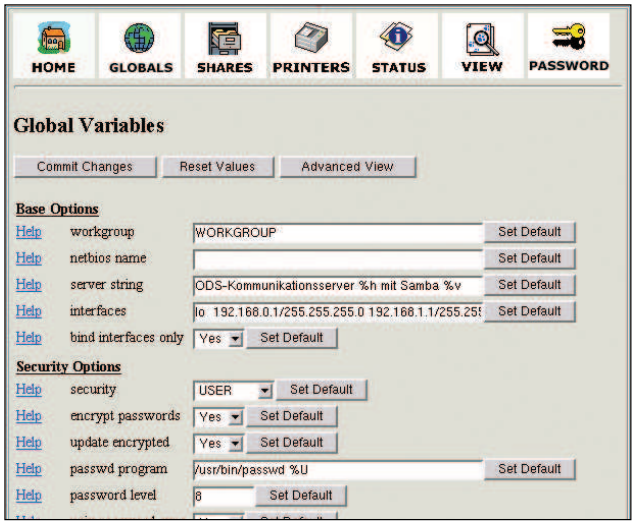
Példa a /etc/smb.conf állományra

```
[global]
         netbios name = linux
         workgroup = munkacsoport
         security = user
         printcap name = /etc/printcap

[homes]
         comment = Sajat konyvtarak
         read only = No
         browseable = No

[printers]
         comment = Nyomtatok
         print ok = Yes
         browseable = No

[cdrom]
         path = /mnt/cdrom


[dolgok]
   comment = Titkos dolgok
   path = /mnt/dolgok
   valid users = bela, jani
   public = no
   writable = yes
   printable = no
```

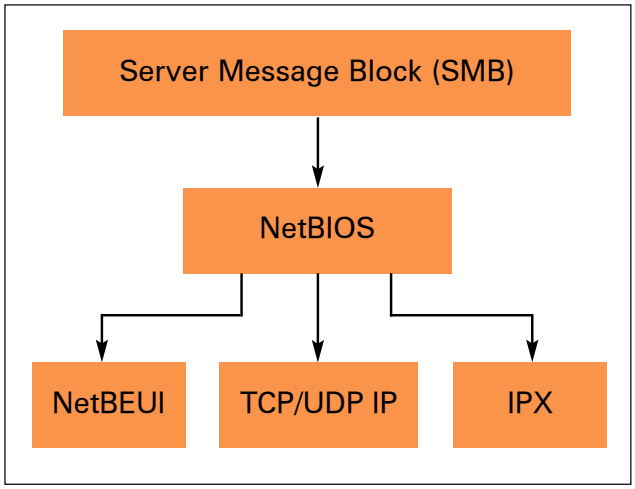
Mi is az a Samba?

A Samba az egy olyan programcsomag Linuxra (és más Unix-rendszerekre), ami lehetővé teszi az SMB protokoll használatát. Segítségével könnyedén használhatjuk a windowsos gépek által megosztott erőforrásokat, és mi is elérhetővé tehetjük számukra a fájlrendszerünket és a nyomtatónkat. A Sambát azonban tiszta unixos környezetben is használhatjuk, ha például valami okból ki szeretnénk vele váltani az NFS protokollt.

A Samba rendkívül széles körű szolgáltatásokat nyújt. Nem-



A képen a SWAT, a Samba webes beállítófelülete látható. Segítségével könnyedén, barátságos környezetben végezhetjük el a beállításokat, akár egy távoli gépről is. Komoly hátrány azonban, hogy a SWAT nem használ semmiféle titkosítást, így jelszavunkat bárki lehallgathatja. Tehát csak olyan hálózaton használjuk, ahol biztosan nincsenek „hallgatózó fülek”



A Server Messge Block (SMB) protokoll a NetBIOS-ra épül. Láthatjuk, hogy az SMB számára érdektelen, hogy maga az átvitel milyen protokoll segítségével zajlik, legyen szó TCP/IP-ről vagy IPX-ről

csak SMB-kiszolgálót és -ügyfelet tartalmaz, hanem NetBIOS-kiszolgálóként működik és némi parancssori NT alapú felügyeleti lehetőséget is kínál.
A fájl- és nyomtatómegosztás az `smbd` nevű démon feladata. Az `nmbd` a NetBIOS névfeloldásáért felelős démon. Az `smbclient` egy FTP-ügyfélhez hasonló alkalmazás, ennek segítségével érhetjük el a más gépek által megosztott állományokat.

A Samba beállítása

A Samba beállításait a */etc/smb.conf*, egyes terjesztéseknél a */etc/samba/smb.conf* állomány raktározza. Itt sorolhatjuk fel megosztani kívánt erőforrásainkat is. Az állomány felépítése nem túl bonyolult: az erőforrás nevét szögletes zárójelbe írjuk, majd utána felsoroljuk a kapcsolóit.
Létezik azonban három különleges csoport. Az első a `[global]`, ahol a kiszolgáló általános, illetve alapértelmezett beállításait adhatjuk meg. A `[homes]`, ami ha jelen van, akkor a bejelentkezett felhasználók önműködően hozzáférhetnek a saját *home* könyvtárukhoz. A `[printers]` hasonló az előbbihez, az eltérés csak annyi, hogy ez a géphez csatlakoztatott nyomtatókra vonatkozik (pontosabban a felhasználó használhatja azokat a nyomtatókat, amelyeket a */etc/printcap* állományban már előzőleg megadtunk).
Az *smb.conf*-ban az erőforrásokhoz megadható összes kapcsoló ismertetésére ehelyütt nem tudunk kitérni, mivel ezekkel egy egész könyvet meg lehetne tölteni. Ezért most csak azokra térünk ki, amelyek feltétlenül szükségesek ahhoz, hogy erőforrásainkat megoszthassuk. Akik mélyebben meg szeretnének ismerkedni a Samba beállításának rejtelmeivel, azoknak azt javasoljuk, hogy a ➲ http://hu.samba.org/samba/docs/man/smb.conf.5.html címen olvassák el a leírást.
Az *ábrán* egy lehetséges beállítást láthatunk a Sambához. A saját könyvtárakon és a nyomtatón kívül megosztjuk a CD-ROM-ot és a */mnt/dolgok* nevű könyvtárat. Most megnézünk néhányat a példában látható kapcsolók közül, a `[global]` csoporttal kezdve.
A netbios name a Samba kiszolgáló nevét jelenti, amellyel majd az ügyfelek hivatkoznak rá. Ha ide nem adunk meg semmit, akkor ez az érték meg fog egyezni a gép DNS nevével. A workgrouphoz értelemszerűen a munkacsoport nevét kell írnunk.
A security kapcsolónál a használni kívánt biztonsági szintet adhatjuk meg. A korábban felsorolt két szint mellett (user és share) használhatjuk még a server és domain értékeket, ezek kis különbségekkel megegyeznek a userrel. (Az eltérés abban nyilvánul meg, hogy ilyenkor nem maga a linuxos gép azonosítja a felhasználót, hanem egy másik SMB-kiszolgálóhoz vagy a Windows NT-tartományhoz fordul. Hogy ez pontosan miképpen történik, arról kimerítő útmutatást kaphatunk a leírásból).
A browsable azt mondja meg, hogy az ügyfelek láthatják-e a hálózaton jelenlévő osztott erőforrásokat mutató listában.
A read only-val értelemszerűen azt állíthatjuk be, hogy az adott erőforrás csak olvasható-e vagy írható is (de használhatjuk ennek a fordítottját, a `writeable` kapcsolót is). A `read only` a nyomtatók esetében azt jelenti, hogy a felhasználó nem írhat közvetlenül a nyomtatandó anyagokat tartalmazó nyomtatási gyűjtőkönyvtárba (spool).
A `public` megegyezik a `guest ok` kapcsolóval, és ha az értéke igaz, akkor nem kell jelszót megadni (vagy érvényes felhasználó név–jelszó párost) az eszköz eléréséhez. A `valid users` segítségével lehetőségünk nyílik megmondani, hogy egy adott erőforrást milyen felhasználók érhetik csak el (illetve az `invalid users`-szel azokat, akik nem érhetik el).
A `printable` vagy `print ok` azt jelenti, hogy a nyomtatóra küldött állományok nyomtatásra is kerüljenek. Ezt mindenképpen érdemes bekapcsolni.
Láthatjuk, hogy az SMB-kiszolgáló beállítása annyira nem ördöngös művelet. Mindenesetre van egy könnyebb módja is a beállításnak: a `swat` nevű kisalkalmazás használata, ennek segítségével weben keresztül is elvégezhetjük a beállításokat.
A `swat`-et is az `inetd` indítja, és a 901-es kapunk üzemel.
A `swat`-et azonban óvatosan használjuk, mert semmiféle titkosítást nem használ, így a jelszavunkat könnyedén lehallgathatják. Ezért akik a webes beállítás hívei, azok inkább a Webmint használják, a Samba összetevővel.

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.

Laptopnet

Hordozható számítógépek hálózati beállításainak könnyed módosítása.

Adva van egy hordozható számítógép, amit több helyszínen, különféle hálózati beállításokkal szeretnénk használni, és ehhez szükségünk van valami olyasféle segédprogramra, amivel könnyedén beállíthatjuk (netán önműködővé tehetjük) megváltozott hálózati beállításaink betöltését.
Nagyon sok segédeszköz áll a vállalkozó szellemű felhasználók rendelkezésére. A számos lehetőség közül természetesen igyekeztem a legjobbat megtalálni – nézzük, mik a lehetőségek Debian/GNU Woody alatt:
`divine` – önműködően felismeri a hálózatot.
`guessnet` – kitalálja, melyik hálózathoz is kapcsolódunk.
`intuitively` – önműködő IP-beállító program.
`laptop-netconf` – hálózatfelismerő és beállítóprogram laptopokhoz.
`switchconf` – laptophálózati beállításváltás.
`whereami` – önműködően felismeri és beállítja a hálózatot.

```
#!/bin/sh

if [ $1 = office ];
  then
       echo 'Megint dolgozunk :-(((';
       echo '';

       #Irodai hÆl zat beÆll tÆsa
       /etc/init.d/networking stop;
       ln -sf /etc/network/interfaces.office
       ➥/etc/network/interfaces;
       ln -sf /etc/resolv.office
       ➥/etc/resolv.conf;
       /etc/init.d/networking start;

  elif [ $1 = home ];
       then
            echo 'Otthon vagyunk vØgre :-)))';
            echo '';

            #Otthoni hÆl zat beÆll tÆsa
            /etc/init.d/networking stop;
            ln -sf /etc/network/interfaces.home
            ➥/etc/network/interfaces;
            ln -sf /etc/resolv.home
            ➥/etc/resolv.conf;
            /etc/init.d/networking start;

       else

  echo 'A program hasznÆlata: net
       ➥kapcsol  (office, home...)';

fi
```

A fenti lista nem teljes, csak azokat a programokat tartalmazza, amiket kipróbáltam. A kipróbálás, ugye, mindig úgy kezdődik, hogy a kiválasztott programot telepítjük, beállítjuk és kipróbáljuk, először, másodszor, harmadszor… Miután az ezzel eltöltött időmennyiségen feldühödtem, megírtam a saját parancsfájlomat. Ez a munka körülbelül öt percet vett igénybe, nem kellett különféle beállítófájlokban turkálnom, kitalálni, hogy ki mit hová és miért tett abban a bizonyos fájlban, ami az egyedüli üdvözítő megoldást tartalmazná.

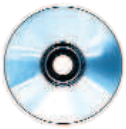
A megoldás

A hálózati beállításokat néhány fájlban tárolt adat határozza meg, a két legfontosabb fájl a */etc/network/interfaces* és a */etc/resolv.conf*. Ezekben kell megváltoztatnunk az adatokat, újraindítani a hálózati csatolónkat, és máris használhatjuk a hálózatot.
Mindkét fájlból másolatot készítettem, egyet az otthoni hálózathoz, egyet pedig a munkahelyi hálózathoz. A kapott fájlok az *interfaces.home*, *resolv.home*, illetve az *interfaces.office* és *resolv.office* nevet kapták (mindegyik fájl az eredeti fájlnak megfelelő könyvtárban került elhelyezésre). Ezután leállítottam a hálózatot a gépen, és letöröltem az eredeti fájlokat. Elkészítettem egy parancsfájlt (*listánkon*), ami annak megfelelően, hogy mit kap kapcsolóként, átállítja a fájlok hivatkozását. Tudom, lehetne még csinosítgatni, de így is működik, ezért győzött a lustaság (legalábbis nálam). A parancsfájl neve `net` lett, így ha kiadom a `net office` parancsot, akkor a munkahelyi beállítások élednek fel, ha pedig a `net home` utasítást gépelem be, akkor az otthoniak. Ha rosszul adjuk meg a kapcsolót, hibaüzenetet kapunk. Tudom, hogy nagyon „egyszerűen" oldottam meg ezt a feladatot, de az a lényeg, hogy működik, kitehetek egy ikont az asztalomra, és erre kattintva máris élvezhetem munkám gyümölcsét.

A jövő

Ha sikerül rávennem magamat, akkor elkészítem ennek a programnak az önálló, működésre képes változatát is, ami rendszerünk elindulásakor önműködően felismeri a hálózatot. Tudom, most sokan azt mondhatják, hogy „minek, van már ilyen program"; de én nem azért készíteném, hogy legyen még egy, hanem azért, mert az enyém! A fent említett programokat nem sikerült megszeretnem, vagy túl sok időbe került a beállításuk, vagy hol működtek, hol nem, teljesen akadozó volt a teljesítményük, ezért is írtam ezt a kis programocskát.
Jó szórakozást, és ha esetleg valaki kedvet érez a program csinosítgatásához, mondjuk megcsinálná helyettem az automatikus hálózat felismerés részét, az bátran tegye meg.
A kérésem csak annyi, hogy küldjön nekem is egy példányt!

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



Amit néha saját kezűleg kell megtennünk

A KDE Desktop Sharing és az rdesktop beállítása és használata.

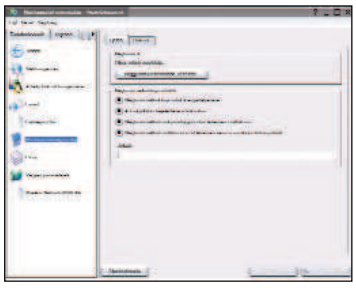
François, ellenőrizd, légy szíves, hogy a hálózati elosztó csatlakoztatva van-e. Merci, mon ami. Most pedig figyeld, hogy csatlakozom a borospince keleti szárnyában lévő terminálhoz. Mint látod, innen fentről is teljes körű hozzáférésem van a rendszerhez. Ellenőrizhetem a munkaasztalt, programokat indíthatok vagy telepíthetek és így tovább. Ha most lent lennél, François, mindketten használhatnánk a munkafelületet. Hogyhogy mi a jó abban? Előfordul, François, hogy egy távoli irányítási lehetőség mindent megér. Ha meg tudod osztani munkafelületedet egy másik felhasználóval, vagy szükség esetén át tudod venni valaki más rendszerének az irányítását, rengeteg időt takaríthatsz meg a folyamat irányításával vagy akár csak azzal, hogy megfigyelheted, mi történik. Tegyük fel, hogy egy bizonyos pincér nemrég telepítette a Linux-rendszerét, és néha szüksége van egy kis útmutatásra. Mais non, François, ez csak egy feltételezés. Mindenki tudja, hogy te mennyire értesz hozzá. Na, ne háborogj már!
Á, mes amis! Isten hozott titeket Chez Marcelnél, a páratlan Linux-konyha, a figyelmes kiszolgálás és a kitűnő borok otthonában. Üljetek le, helyezzétek magatokat kényelembe. François, miért nem sietsz már a pincébe? Azt hiszem, valamilyen olasz bor illene a mai alkalomhoz. Hozd fel a 97-es Brunello di Montalcinót a vendégeinknek!
Éppen azt magyaráztam François-nak, hogy adódhatnak olyan helyzetek, amikor tennivalónk elvégzéséhez a legmegfelelőbb eszközt egy távoli hozzáférés nyújtja. Talán a kényelmes irodai környezet adja ehhez a legjobb ösztönzést: a munkafelület megosztásával a rendszergazda anélkül képes kezelni az önálló munkaasztalokat, hogy felkelne a kedvenc karosszékéből. Meg kellene mutatnod egy felhasználónak, hogyan tud kitenni egy ikont az asztalára? Csatlakozz a gépére, és hagyd, hogy végignézze. Segélyhívás érkezett, hogy segíts értelmezni egy hibaüzenetet? Csatlakozz a felhasználó rendszerére, és ismételtesd el vele a hibát kiváltó műveletsort, eközben figyelemmel kísérheted, mi történik. A lehetőségek korlátlanok.
Az angliai Cambridge-ben működő AT&T Laboratories kitűnő, több felületen működő távfelügyelő programcsomagja a VNC. A témáról a Linux Journal honlapján olvasható egy írásom, a ➲ http://www.linuxjournal.com/article/5333 címen. A VNC-vel az a gond, hogy semmilyen eszközt nem nyújt az X-felület (`$DISPLAY:0`) vezérlésére, így nincs mód arra, hogy egy felhasználó grafikus asztalának irányításával helyrehozzunk egy hibát, vagy valamit ezzel a módszerrel szemléltessünk. Ekkor siet a segítségünkre a ***Tim Jansen*** alkotta KDE Desktop Sharing (KDE munkafelület-megosztó). A programcsomag szerepel a KDE 3.1 hálózati eszközeinek listáján. A KDE 3.0 felhasználói szintén élvezhetik a program nyújtotta előnyöket, ehhez csupán Tim weboldalát kell felkeresniük (➲ http://www.tjansen.de/krfb) és letölteni a legfrissebb forráskódot. (47. CD Magazin/Fogado)
A csomag fordítása a szokásos ötlépéses kicsomagoló és fordító eljárás egy kicsit módosítva, ami csak annyit tesz, hogy a csomag az egyszerű `gzip` módszer helyett a `bzip2` tömörítést használja.

```
tar -xjvf desktopsharing-0.7.tar.bz2
cd desktopsharing-0.7
./configure --prefix=path to KDE
make
su -c "make install"
```

A fent látható előtagmegadást azért ajánlom, mert a ***KDE Desktop Sharing*** a KDE vezérlőtóközpontjának (***Control Center***) egységeként fog működni, így KDE-példányunkkal nem árt tudatni a program tartózkodási helyét. A programot kipróbáltam Red Hat, Mandrake és SuSE rendszeren is. A Red Hat és a Mandrake KDE-könyvtárai a ***/usr*** könyvtárban foglalnak helyet, míg SuSE esetén a ***/opt/kde3*** a tartózkodási helyük. Mielőtt a program használatára térnénk át, szeretnék még egy lehetséges zavarforrást megszüntetni. A különböző Linux-rendszercsomagok néha nem a megszokott KDE-menüelrendezést választják, vagyis a telepítőprogram nem találja a dolgokat a feltételezett helyükön. Példa erre a Mandrake 9.0 asztala, aminél a telepítő ugyan zökkenőmentesen lefutott, de a ***Desktop Sharing*** menüpont mégsem jelent meg a ***Control Center***-ben. Ha az okot tudjuk, a javítás már egyszerű. Csupán annyit kell tennünk, hogy a ***desktopsharing*** könyvtárából átlépünk a ***krfb/kcm_krfb*** könyvtárba, és a ***KDE Control Center*** bejegyzését az alábbi (egyetlen) paranccsal külön telepítjük:

```
/usr/bin/install -c -p -m 644
➥kcmkrfb.desktop/usr/share/applnk-mdk/
➥Configuration/KDE/Network/
```

Ismét hangsúlyozom, hogy mihelyt a programcsomag belekerül az aktuális KDE-változatba (a 3.1-től kezdődően), ezek a telepítési gondok megszűnnek. A fordítás, a telepítés és a szükséges módosítások elvégzése után lépjünk ki a KDE-felületből, mielőtt továbblépnénk. Míg a munkaasztalunk újraindul, használjuk ki az időt arra, hogy ennek az igen kitűnő bornak a zamatát megízleljük.



1. kép A KDE Desktop Sharing beállítása a Control Centerből

Az ügyfélgép távoli elérésre való felkészítését kezdjük azzal, hogy a programindító menüből (a bal alsó sarokban lévő nagy K) elindítjuk a ***KDE Control Center***-t. Ezt megtehetjük úgy is, hogy a héj parancssorába beírjuk a `kcontrol &` parancsot. A `kcontrol` elindulása után kattintsunk a bal oldali, menüsávon lévő ***Hálózat*** (***Network***) ikonra, és válasszuk a Desktop Sharing menüpontot, mint az *1. kép* mutatja. Látható, hogy a jobb oldalon két ablakfül jelenik meg, az egyik az ***Hozzáférés*** (Access), a másik pedig a ***Hálózat*** (Network) feliratot viseli.

A ***Hálózat*** fül kevesebb magyarázatot kíván, ezzel kezdem. Kattintsunk a fülre, ezzel lehetőségünk nyílik arra, hogy felülbíráljuk a ***Port hozzárendelése*** (Assign port) alapértékét. Ha nem adunk meg más értéket, a ***KDE Desktop Sharing*** az 5900-as kaput használja, de a jelölőnégyzetet átállítva tetszőleges kapu megadására nyílik lehetőségünk. Lépjünk tovább a ***Hozzáférés*** fülre. A ***Meghívás módja*** (Invitations) a távoli asztalhoz való hozzáférési jogosultság megadásának a módját jelenti, de lehetőségünk van az ezt mellőző kapcsolat engedélyezésére is. Úgy gondolom, mes amis, hogy nem kell külön ecsetelnem ez utóbbi megoldásnak a biztonsággal kapcsolatos vonatkozásait, ezért hadd térjek rá mindjárt a meghívás létrehozásának és kezelésének témájára.

2. kép
Személyre szóló meghívás létrehozása a KDE Control Session számára

3. kép Csatlakozási kérés érkezik

Elsőként kattintsunk a ***Meghívások létrehozása és kezelése*** (Create & Manage Invitations) gombra. A megjelenő ablak két lehetőséget ajánl: létrehozhatunk egy ***Új személyre szóló meghívás***-t (New Personal Invitation) vagy ***Levélmeghívás***-t (New E-mail Invitation).
Biztonsági megfontolásból a meghívási folyamat csak egy órára engedélyezi a hozzáférést, vagyis a ***Desktop Sharing*** önműködően tárolja a jelszót és lejárati időt a munkafolyamat számára. A kapcsolat létrehozásához szükséges IP-cím szintén itt látható. Sem a jelszó, sem a lejárati idő hatálytalanítására nincs lehetőség. Arról magunknak kell gondoskodnunk, hogy a kapcsolódáshoz szükséges adatokat az itt látható formában továbbítsuk a kapcsolódni szándékozó személynek. Ha ez megtörtént (vagy feljegyeztük őket), kattintsunk a ***Bezárás*** (Close) gombra.
A másik lehetőség az elektronikus levélen keresztül történő meghívás. Ennek az az egyetlen buktatója, hogy az egyórás időtartam alatti hozzáférés módját levélen keresztül küldjük el. A rendszer fel is hívja a figyelmünket a szöveges levelek Világhálón való küldésénél, szorgalmazva az ilyen levelek titkosítását. Kattintsunk a ***Tovább*** (Continue) feliratra, ezzel túllépünk a figyelmeztetésen, és a KMail üzenete jelzi, hogy készen állunk a küldésre. Ha a meghívásunkra nem érkezik válasz, az egy óra múlva érvényét veszti. Egyébként a meghívásokat az alábbi paranccsal is kezelhetjük:

```
krfb &
```

Mielőtt továbblépnénk, kattintsunk a ***Bezárás*** gombra, hogy magunk mögött hagyjuk a meghívásos kapcsolatengedélyezést, és vessünk egy pillantást a hozzáférés engedélyezésének meghívás nélküli módjára. Ha egy levélben elküldött meghívás is fontos biztonsági kérdéseket vet fel, akkor egy szélesre tárt folyamatos meghívásjoggal szólaltatja meg a vészcsengőt. Egy irodai környezetben azonban ez lehet a hozzáférés biztosításának a legésszerűbb módja. Ha engedélyezzük a meghívás nélküli kapcsolatot, akkor is jelszót kell rendelnünk a kapcsolódáshoz. Lehetőségünk van arra is, hogy a kapcsolat létesítése előtt azt a fogadó gépen jóvá kelljen hagyni (Confirm). Azt ugyancsak eldönthetjük, hogy az ilyen kapcsolatoknak engedélyezzük-e az irányításátvételét.

4. kép Windows Terminál Server-féle csatlakozás az rdesktop segítségével

A kapcsolódáshoz bármelyik VNC-ügyfélprogramot használhatjuk, de az egyszerűbb megoldás az, ha az irányítógépre is a ***KDE Desktop Sharing*** kerül. Felülete könnyen kezelhető és vonzó megjelenésű. Az ügyfél elindítását a K alatt megnyíló ***Internet*** menüpontból vagy közvetlenül a parancssorból kezdeményezhetjük:

```
krdc &
```

Miután létrehoztunk egy olyan élő meghívást, amelyben beállítottuk a jóváhagyás kérését, egy csatlakozni kívánó távoli ügyfél hatására egy figyelmeztető üzenetet jelenik meg, ami felkínálja a kapcsolat engedélyezésének, illetve elutasításának lehetőségét (*3. kép*).
A kérés elfogadása után a távoli felhasználónak be kell írnia a jelszót, a rendszer tálcáján pedig megjelenik egy minket figyelő szép kék szem.
A program egyik legjobb tulajdonsága, hogy a csatlakoztatott munkafelületet szinte bármilyen méretűre beállíthatjuk. Állítsuk be az ablakunk méretét, majd kattintsunk a nagyító ikonjára. Különösen bizarr látványban lehet részünk akkor, ha a saját gépünkön hozunk létre egy meghívást, és ehhez kapcsolódunk. Hatására ugyanis egy, az asztalunk egymásba ágyazott végtelen sorozatát kapjuk, ami ahhoz hasonló látványt nyújt, amikor két egymás felé néző párhuzamos tükör közé állunk.
Nagyon izgalmas, amikor azt látjuk, hogy a Linux mennyivel közelebb hozhat egymáshoz bennünket, non? Ez akkor is igaz, amikor más operációs rendszert futtató gépekkel is dolgozunk. Hadd mutassam meg ezt is!
Egy másik figyelemreméltó távfelügyelő program a ***Matt Chapman*** megalkotta rdesktop. Az ötletet a következő szolgáltatta: előfordulhat, hogy rendszeres tennivalónk akad egy Windows 2000-et futtató gépen. Ha az adott gép a Win2K Terminal Serverrel való csatlakozást igényli, akkor ehhez többé nem kell leállítanunk a Linux-rendszerünket.

Röviden, az rdesktop egy a GPL felhasználói szerződés alá tartozó Windows (NT/2000) alatt futtatható Terminal Server ügyfélprogram, ami annyit jelent, hogy az RDP-t (Remote Desktop Protocol) használja. Amennyiben a programot használni szeretnénk, töltsük le a forrást a ➲ http://www.rdesktop.org címről (47. CD Magazin/Fogado), majd fordítsuk le a szokásos ötlépéses kicsomagoló-fordító eljárással:

```
tar -xzvf rdesktop-1.1.0.tar.gz
cd rdesktop-1.1.0
./configure
make
su -c "make install"
```

Az egész nem tarthat tovább néhány másodpercnél. A telepítés után a program a következő paranccsal indítható:

```
rdesktop -u Administrator -p PaSsWoRd
192.168.22.212
```

A `-u` kapcsoló a Windows-kiszolgáló felhasználóját adja meg, a `-p` pedig a jelszót. Vessetek egy pillantást az *4. képre*, amelyen KDE 3.1 munkaasztalom látható, amint épp az rdesktopot futtatom egy távoli Windows 2000 kiszolgálóra csatlakozva. Mint láthatjátok, mes amis, egy hálózati kapcsolat és a Linux-rendszeretek segítségével máris legyőztük a távolságokat. Mintha csak ott lennénk a helyszínen.

Mon Dieu, az idő viszont gyorsan elszállt. Mivel a távoli csatlakozás alkalmatlan arra, hogy egy pohár bort töltsek nektek, ezt még azelőtt kell megtennünk, hogy François-val bezárnánk éjszakára az éttermet. François, lennél olyan szíves, és még utoljára teletöltenéd a vendégeink poharát? Igyunk egymás egészségére a következő viszontlátásig.
A votre santé! Bon appétit!

Linux Journal 2003. április, 108. szám

Marcel Gagné (mggagne@salmar.com)
Mississaguában, Ontario államban él. Ő a szerzője a Kiskapu kiadásában szeptemberben megjelent Linux-rendszerfelügyelet (ISBN 96-9301-40) című könyvnek (jelenleg is egy könyvön dolgozik).

KAPCSOLÓDÓ CÍMEK

KDE Desktop Sharing ➲ http://www.tjansen.de/krfb
rdesktop ➲ http://www.rdesktop.org
VNC (Virtual Network Computing)
➲ http://www.realvnc.com
Az X0rfbserver honlapja ➲ http://forum.hexonet.com
Marcel borlapja ➲ http://www.marcelgagne.com/wine.html

Sablonkezelés Smarty módra (2. rész)

Múlt hónapban, ha nem is alaposan, de megismerkedtünk a Smarty sablonrendszerrel. Ezt folytatva most egy kicsit mélyebbre ásunk…

Már egy hónapja, hogy tudjuk, miért is jó nekünk sablonokkal dolgozni. Tudással bírunk arról, hogy milyen módon kell különféle változóértékeket csempészni a sablonba, vagy akár mindjárt tömböket listázni. Azzal is tisztában vagyunk, hogyan is adjunk át adatokat a sablonjainknak PHP-ből. Ez az alaptudás az, amit egy sablonkezelőnek tudnia kell, ennyivel akár el is boldogulhatnánk. Valójában a Smarty még csak itt kezdődik. Lássuk, miféle hókuszpókuszokat tudunk még vele művelni!

Feltételkezelés

A feltételkezelés sablonmotorunknak az egyik olyan képessége, amit nem feltétlenül helyesel mindenki, hiszen a feltételkezelésnek igazából a logika részének kellene lennie, legalábbis ha összetett feltételek alapján való döntésről van szó. A már régóta jól ismert `if-elseif-else` a sablonokban történő hármas bevetése néhol valóban hasznos lehet. A legegyszerűbb esetben egy sima feltételről van szó:

```
{if $be_van_lepve == true}
<a href="kilep.php">KilØpØs</a>
{/if}
```

Ennél jóval bonyolultabb feltételek is megadhatók és zárójelek használata is lehetséges. Bár itt jegyezném meg, hogy a legtöbb esetben nem túl szerencsés a zárójelek alkalmazása. A bonyolultabb feltételek elemzése nem a sablon feladata, azt már logikai (PHP-kód) szinten el kell dönteni, és egy megfelelő változóban csak az eredményt közölni. Az egyes `if`, `elseif` és `else` ágak a következő ág megnyitásáig, avagy a `{/if}` címkéig tartanak.

Értékmódosítók

Sorozatunk előző részében bemutattam, hogyan lehet egyszerű értékbehelyettesítéseket végezni. Emlékeztetőül idézzük vissza, hogy ez milyen módon is zajlik:

```
Postafi kodban {$uzentek_szama}  zenet talÆlhat .
```

Előfordulhat, hogy sablonokkal dolgozó munkatársunkban felmerül az igény az iránt, hogy a kapott adatokat esetleg szeretné kicsit csinosítgatni, feljavítani. Vagy zavarja, hogy a szépen megtervezett üzenetdobozba olyan szöveg is kikerülhet, ami „szétnyomja” azt. Ilyenkor természetesen a programozót lehet nyaggatni, hogy márpedig ezeket a módosításokat kéri a kapott értékekben. Ám jobban belegondolva ezek a dolgok inkább a megjelenítési réteghez tartozó feladatok, semmint a programlogika részei. Már csak azért is, mivel egy oldalon ugyanaz az érték akár többször több formában is megjelenhet. Ráadásul nem tudni, hogy ami ma éppen csupa nagy betűvel tetszik a megrendelőnek, az másnap nem kellene-e mégis kisbetűkkel neki. A legjobb megoldás tehát a sablonban ezeket a megjelenítési formákat megadni (szenvedjen vele a grafikus, illetve az arculatfelelős). Lássuk, hogyan is oldható meg!
A helyzet kulcsa a | (csővezeték) jel, ezzel elválasztva lehet egy vagy akár több értékmódosítót is egy értékbehelyettesítéshez hozzárendelni. Lássuk példával is szemléltetve:

```
{$megnevezes|upper|spacify}
```

Itt tehát a „megnevezes” néven futó Smarty-érték kerül kiíratásra, ámde előtte két művelet is végrehajtódik rajta. Egyrészt egy `upper` értékmódosító hatására csupa nagybetűvel fog a szöveg kikerülni, a `spacify` hatására pedig szóközök kerülnek a betűk közé.
Az adatmódosító eszközök közül több értékekkel is ellátható. Ezek megadása közvetlenül az adatmódosító nevének megadása után lehetséges, kettősponttal elválasztva. Helyenként több érték is megadható, ilyenkor ezeket is kettősponttal kell egymástól elválasztani. Mindjárt ilyen lehetőségeket rejt maga a `spacify` is, ugyanis a szóközkarakterek beszúrása csak az alapértelmezett viselkedése. Ehelyett azonban bármilyen más karakter vagy karakterfüzér is beilleszthető a szöveg betűi közé, például az alábbi sabloncímkék alkalmazásával:

```
{$telepules}
{$telepules|spacify}
{$telepules|spacify:".."}
```

Ennek hatására ilyesféle eredményeket kapunk:

```
SzØkesfehØrvÆr
S z Ø k e s f e h Ø r v Æ r
S..z..Ø..k..e..s..f..e..h..Ø..r..v..Æ..r
```

A Smarty beépített módosítói

Három külön csoportra osztottam a Smarty beépített módosítóit. A legegyszerűbbek kerültek az egyik csoportba, amelyek apróbb formai átalakítást végeznek. Ezeknek paraméterezési lehetőségük sincs. A következő és egyben legnagyobb csoportba azok az értékmódosítók kerültek, amelyek különféle értékeket várnak. Ezek egy része kötelezően elvár bizonyos adatokat, mások nem – szükségességét mindenhol helyben jelzem. A harmadik csoport logikailag igazából nem tartozik az értékmódosítók közé, de megvalósításuk tekintetében ide sorolhatók. Ezek a kapott szöveg átalakítgatása helyett különféle, az adott szövegre jellemző számadatokra cserélik ki őket.

Egyszerű, érték nélküli módosítók

- `capitalize`: minden egyes szó kezdőbetűjét nagybetűssé alakítja.
- `lower`: a teljes szöveget kisbetűsíti.
- `upper`: a teljes szöveget nagybetűsíti.
- `nl2br`: a PHP `nl2br()` függvényének megfelelően a soremelés karaktereket `<br />` XHTML címkékre cseréli.

Paraméterezhető értékmódosítók

- `cat`: az egyetlen kötelező értékben megadott szöveget a behelyettesített szöveg végéhez fűzi. Elsőre értelmetlennek tűnik, hiszen ez a szöveg kerülhetne egyszerűen a sablon állandó részébe is. Így viszont további adatmódosítások is végezhetők a teljes összefűzött szövegen.
- `date_format`: a PHP `strftime()` függvényének megfelelő formában megadott formázás szerint jeleníti meg a dátumot, illetve az időt. Ez a forma az első értékben adható meg. Ha nincs megadva, az alapértéke: `%b %e, %Y`. Második értékként megadható egy szöveg, ami akkor jelenik meg, ha az eredeti érték üres volt. Nem kötelező élni vele.
- `default`: amennyiben a kérdéses érték üres, helyette a `default` egyetlen kötelezően megadott értékének szövege jelenik meg. Létező érték esetén azt érintetlenül hagyja.
- `escape`: egyetlen, egyben kötelezően megadandó értéke van, ami azt határozza meg, hogy milyen kódolással legyen az adott szöveg „levédve".

  A választási lehetőségek

  - `html`: a `htmlspecialchars()` PHP-függvénynek megfelelően alakítja át a szöveget. Mindezt bekapcsolt `ENT_QUOTES` értékkel, ami annyit tesz, hogy a `< (<) > (>) & (&)` és az idézőjel (") karakterek mellett az egyszeres idézőjelet (aposztrófot) is átkódolja ('). Jól használható HTML-forráslista megjelenítésére.
  - `htmlall`: az előzőhöz hasonlóan működik, de a fent felsoroltakon túl minden olyan karaktert átkódol, amiknek létezik HTML-egyed megfelelőjük.
  - `url`: a PHP `urlencode()` függvényét alkalmazza az adott szövegre.
  - `quotes`: az idézőjeleket visszaperjellel (\) védi le.
  - `hex`: a `%xx` formára kódol minden karaktert, ahol az „xx" az adott karakter kódja tizenhatos számrendszerben megadva.
  - `hexentity`: az előzőhöz hasonlóan a tizenhatos számrendszerben dolgozik, de szabványos HTML-hexaegyedeket hoz létre, például: `&#x6d;`
  - `javascript`: a következő cseréket hajtja végre: \-ből \\, '-ből \', a soremelés és kocsivissza karakterekből pedig egyöntetűen `\r` lesz.
- `indent`: a szövegek igazítására készült módosító. Minden sor elejére adott számú szóközt szúr be. A második értékben a szóközkarakter is lecserélhető.
- `regex_replace`: két kötelező értékkel bír, az első a keresési mintát, míg a második a csereszövegeket tartalmazza. Teljes mértékben megfelelnek a PHP `preg_replace()` függvénye első két értékének.
- `replace`: egyszerűbb cserékre való. Az `str_replace()` PHP-függvénynek felel meg.
- `spacify`: a példában már megismert módosító. Alapesetben minden két karakter közé egy szóközt helyez el, de megadható helyette más, akár többkarakteres szöveg is.
- `string_format`: egy értéke van, ez a PHP `sprintf()` függvényének megfelelő alakban megadott formázás.
- `strip`: minden többszörözött szóköz, tabulátor és újsor karaktersorokat egyetlen szóközre cserél. Értékként más karakter vagy karaktersor is megadható.
- `strip_tags`: az összes HTML-címkétől megszabadítja a szöveget. Egyetlen értékben megadható, hogy logikai értékként a HTML-címkék helyén hagyjon-e szóközt. Ha nem adjuk meg, hagy.
- `truncate`: hosszú szövegek csonkítására való. Az első értéke sem kötelező, de itt adható meg az a hossz, ami felett vágni szeretnénk. Alapértéke: 80. Második értékként a vágás helyén megjelenő „..." karaktersorozat cserélhető le esetlegesen másra. Harmadikként megadható, hogy a szavak közepén is vágjon-e. Ennek alapértéke: `false`.
- `wordwrap`: szövegtördelés. Három nem kötelező értéke van. Az elsőben azt módosíthatjuk, hogy hányadik karakteroszlopnál törje a sorokat, ennek alapértéke 80. Másodikként megadható, hogy milyen karakter vagy karaktersorozat legyen beszúrva sortörésként. Ha másképp nem rendelkezünk, ez a `"\n"` soremelés karakter lesz. Harmadikként egy logikai értéket vár, ami eredetileg hamisra (`false`) van állítva, ilyenkor csak szóhatárokon tör. Ha igaz (`true`) állapotba billentjük, pontosan az adott oszlopban fogja megtörni a sorokat, nem törődve azzal, hogy szó közepén jár-e, vagy sem. Példa:

```
{$hosszuszoveg|wordwrap:68:"<br>\n"}
```

Szövegelemző, számszerű adatokat közlő módosítók

- `count_characters`: a szöveg helyett annak karakterekben számolt hosszát adja vissza.
- `count paragraphs`: a szöveg helyett a benne található bekezdések számát adja vissza (a soremeléseket alapul véve).
- `count_sentences`: az előző kettőhöz hasonlatosan számot ad vissza, a szövegben található mondatok számát.
- `count_words`: megszámolja a szavakat, és a kapott számértéket adja vissza.

Egy értékbehelyettesítéshez több módosító is hozzárendelhető, ezeket további csővezeték (|) jelekkel lehet egymástól elválasztva kitenni. Kiértékelésük sorban balról jobbra történik. Például:

```
{$talalt_weboldal_szovege|strip_tags:false
➥|truncate:60}
```

Ilyen módon a keresőoldalon a talált oldalak szövegtartalmának első hatvan karakterét tudjuk kiíratni. Először a HTML-formázó címkéit távolítjuk el, majd a szöveget legfeljebb hatvan karakterre megvágjuk. Ha a két módosítót megcserélnénk, nem a hasznos szöveg hosszára vonatkozóan adnánk felső határt, hanem a nyers HTML-formázásokkal teli szövegre, ami jelen esetben kedvezőtlen.
Igaz, eddig úgy beszéltem az értékmódosító eszközökről, mintha azok csak változóbehelyettesítéskor működnének, ámde ezek más helyeken is hasznosíthatók:

```
{if $allat|upper eq "KACSA"}
```

Saját értékmódosítók készítése

A dolog sokkal kevésbé ördöngös, mint elsőre gondolnánk. Szerencsére a Smarty rendszert a könnyű bővíthetőség is jellemzi. Első lépésként nézzünk bele a Smarty ***plugins*** könyvtárába. Itt számos ***.php***-re végződő fájlt fogunk találni, de ezek közül is most a „modifier" karaktersorral kezdődők fognak minket érdekelni. Nézzünk is bele egybe, legyen ez például a ***modifier.capitalize.php***. A lényeg itt csupán ennyi:

```
function smarty_modifier_capitalize($string)
{
        return ucwords($string);
}
```

A PHP `ucwords()` függvénye pontosan azt teszi, amit a Smarty-féle `capitalize` értékmódosítónak is el kell az adott szövegen végeznie. Ez nem is csoda, hiszen a dolog megvalósítására is ezt a PHP-függvényt alkalmazza. Saját értékmódosító létrehozásához nem is kell más, minthogy létrehozzunk egy ***modifier.módosítónkneve.php*** állományt a ***plugins*** könyvtárban, amibe egy `smarty_modifier_m dos t nkneve` néven futó függvény teszünk. Az értékben kapott szöveggel ezután azt csinálunk, amit akarunk, csak végül ne felejtsük el egy `return`-nel visszaküldeni. Ennyi az egész.

Feldolgozástól védett sablonterületek, elkapható kimenet

Könnyen megeshet, hogy a sablonállományokba egy kis JavaScript- vagy stíluslapkódot tennénk. A gond ezekkel az, hogy sűrűn alkalmazzák a { } jelpárost. Smarty barátunk az ilyen jeleket látva azon nyomban megkísérli értelmezni az ott található karaktersort, de azonnal bele is bukik. Nem kívánatos, hogy ilyen helyeken kutakodjon, de ezt külön meg kell mondani neki. Ilyen kivételes sablonterületek körülhatárolására készült a `{literal}` `{/literal}` címkepáros. Ami ezek közé kerül, azt kedvenc sablonkezelőnk feldolgozás nélkül átengedi. Létrehozhatunk olyan területeket is a sablonon belül, amelyek ugyan feldolgozódnak, de ahelyett, hogy helyben megjelennének, egy sablonváltozóban landolnak. Ezeket a területeket a `{capture}` `{/capture}` címkékkel kell határolni. Megadható, bár nem kötelező, hogy mi legyen a Smarty-változó neve, ahol a feldolgozás után helyet kap a kimenet. Ez alapesetben a `$smarty.capture.default`. Ha a nyitócímkében megadunk `name` értéket, akkor a `default` erre cserélődik le. Egy példával élve: a `{capture name=elteszem}` nyitócímkével ellátott terület kimenete a `$smarty.capture.elteszem` változóban lesz ezekután elérhető.

Beállításfájlok kezelése

Bizony, ilyet is lehet. A Smarty beépített eszközeivel lehetővé teszi számunkra, hogy a projektjeinkhez a különféle beállításokat külön állományokban tároljunk, és ezekből adatokat olvassunk. Ezek a beállításállományok formailag igencsak hasonlatosak egy bizonyos redmondi cég ***.ini***-fájljaira. Minden adat külön sorban adható meg, név = érték formában. A sorokat csak a soremelés karakter zárja, nincs szükség pontosvesszőre vagy hasonló huncutságra. Az értékek idézőjelek (normál vagy egyszeres) közé tehetők, de nem feltétlenül szükséges.
Hogy ezek az adatok bekerüljenek sablonunk vérkeringésébe, a `{config_load}` címkét kell szorgosan alkalmaznunk.
A `file` értékben kell megadnunk, hogy melyik beállítófájl betöltését tűztük ki célul. Amint a beállítófájl betöltésre került, értékei máris felhasználhatók a következő formában:

```
{#ØrtØk_neve_a_beÆll tÆsfÆjlban#}
```

Az ebből a forrásból érkező értékek tehát nem keverednek a PHP-ból érkező adat-hozzárendeléseinkkel. A `{config_load}` további, nem kötelezően megadandó értékekkel vértezhető fel. Az egyik ilyen `section` néven fut, amivel beállítófájlunk meghatározott területeit tudjuk megcélozni. Miket is? Bizony, a beállítófájlok az egyszerű név–érték hozzárendelésen túl egyéb szolgáltatásokat is tudnak nyújtani, ha igény mutatkozik rá.
A beállításainkat tartalmazó állományaink ugyanis területekre (section) bonthatók. Tetszőleges számú terület nyitható egy ilyen fájlban, ehhez a terület elejére kell egy sort beszúrnunk, ahol kapcsos zárójelek között adandó meg a terület neve. Erre a névre hivatkozhatunk tehát a `{config_load} section`

1. lista site_config

```
# ez lesz az ÆltalÆnos ablakfejlØc
pageTitle = "Smarty dem oldal"

[Titok]
oageTitle = "Smarty dem  - titkos oldalak"

[Forum]
oageTitle = "Smarty dem  - f rum"
```

2. lista forum_header

```
{config_load file="site_config.conf"
section="Forum"}
<html>
<head>
  <title>{#pageTitle#}</title>
</head>
</html>
<body>
```

értékében. A terület ott kezdődik tehát, ahol ez a nyitósor áll, és a következő terület megnyitásáig ér a takarója, vagy ennek híján a fájl végéig. Mi lesz azokkal az érték-hozzárendelésekkel, amiket mindjárt a fájl elején, bárminemű területmegjelölés mellőzésével adunk meg? Ezek az értékek a `section` érték meglététől függetlenül minden esetben betöltődnek. Lehetőség nyílik továbbá arra is, hogy egy ilyen mindenható értéket egy területen belül más tartalommal írjunk felül. Lássunk erre egy élő példát, ahol is webes rendszerünk fejlécsorát fogjuk alakítgatni. Az egyes és kettes lista megtalálható a 47. CD Magazin/Smarty könyvtárában.
A fenti példa weboldalunk fórumterületének HTML-fejlécét állítja elő, ennek hatására a böngészőablak fejlécsorában a „Smarty demó – fórum” szöveg bukkan fel. Ha a `section` értéket „Titok”-ra változtatnánk, avagy teljesen elhagynánk, rendre a „Smarty demó – titkos oldalak”, illetve a „Smarty demó oldalak” szöveg kerülne ablakunk tetejére.
Tovább bonyolíthatjuk életünket a `scope` érték megismerésével. Ennek három különféle értéke is lehet: `local`, `parent`, `global`. A `local` annyit tesz, hogy az itt betöltött értékek csak ebben a sablonfájlban fognak hatni. A `parent` esetén nemcsak helyben, de az ezt meghívó (szülő) sablonállományban is hatással lesznek ezek a betöltött értékek. Amennyiben a `global`-t választjuk, mindegy, hol töltjük be adatainkat, azok mindenhol elérhetőek lesznek. De mi is ez a hierarchikus viszony és miféle más sablonokról is van szó? Mindjárt kiderül.

Részsablonok beillesztése

Nem kell sokat dolgozni a Smartyval, hogy az ember felismerje, jó lenne, ha oldalainak azonos (vagy közel azonos) fejléceit és lábléceit nem kellene minden sablonállományban módosítani, ha valami változás áll be, például egy minden oldalon jelen levő menü esetén. A megoldást `{include}`-nak hívják, ez, gondolom, nem nagy meglepetés. Ami mindenképpen elhagyhatatlan számára, az a `file` érték, amelyben a beilleszteni kívánt sablonállomány nevét kell megadni. A beillesztett sablonok további sablonokat húzhatnak be maguk alá és

így tovább. Az így kialakuló szülő-gyermek viszonyok azok, amelyekről a beállítófálj értékek hatókörének taglalásakor értekeztem. A beillesztett sablon szülőjétől minden változóértéket átvesz. Bár a beillesztő utasításunk természetesen {capture} {/capture} címkékkel körülvéve megjelenítés helyett Smarty változóba kényszeríthető, van erre egy rövidebb mód is, mégpedig az {include} assign értékének szorgos alkalmazása. Ekkor ugyanis az itt megjelölt nevű változóba kerül a beillesztett sablon kimenete, a megjelenítés helyett.

Gyorstárazás

Már az nagy sebességnövekedést jelent, hogy a Smarty nem értelmezi minden egyes alkalommal a sablonokat, csak ha azok változtak. Ámde minden egyes lekéréskor lefut az a teljes PHP-kód, amit sablonfordításkor a Smarty hoz létre. Érdemes minden esetben a Smarty gyorstárazási lehetőségeinek kihasználásában gondolkodni, hiszen ezzel jelentős processzoridőt takaríthatunk meg. Igaz, a Smartyval dinamikus tartalmak létrehozása a szokványos feladat, de ezeknek az oldalaknak nem feltétlenül kell percre kész adatot szolgáltatniuk. Esetleg elégséges, ha fél- negyedóránként frissül a tartalmuk, vagy akár még ritkábban. Alkalmazzunk tehát gyorstárat! Mi kell hozzá? Kell egy könyvtár valahol a gépen, aminek nem feltétlenül kell – sőt! – a nyilvános webkönyvtárban lennie, de fontos, hogy a webkiszolgáló mint felhasználó könyvtárakat, fájlokat tudjon benne létrehozni. Ennek a könyvtárnak az elérhetősége adható meg a Smarty objektum $cache_dir változójában. A gyorstárazás bekapcsolása ugyanitt a $caching = true érték hozzárendelésével tehető meg. Ekkor a Smarty a sablon megjelenítésekor csak akkor fogja a sablonból fordított PHP-kódot futtatni, ha nincs hozzá megfelelő tárolt változat a gyorstárban. Ha futtatnia kell, természetesen egyből tárolja is az adott kimenetet. Alapesetben a gyorstár elévülési ideje egy óra. Ezt a $cache_lifetime Smarty objektumváltozón keresztül igazíthatjuk az igényeinkhez. Az időt másodpercekben kell megadni.

Elő- és utószűrők

Két helyen is alakíthatunk még programunkon. Egyrészt a sablonállományok feldolgozásának megkezdése előtt végezhetünk rajtuk szűréseket, másrészt az előállított PHP-kódon is faraghatunk a saját függvényeinkkel. Ehhez mindkét esetben olyan függvényt kell létrehoznunk, ami a szűrendő adatokat kapja első értékében, és az átalakítások után a módosult „szöveget” adja vissza. A függvény meghatározásában második értéknek mindig biggyesszünk oda egy hivatkozott értéket: &$smarty. Természetesen bármiféle más neve is lehet, de ez a „szabvány”. Ha nincs is szükség a szűrőn belül a Smarty objektum elérésére, ezt akkor is oda kell tenni, ugyanis a Smarty objektum meghívásakor feltétlenül átadja.

További tippek és trükkök

Minden assign() hívás egy-egy másolatot készít azokról a változókról, amiben az értéket biztosítjuk számára. Nagyobb tömbök, objektumok esetén ez jelentős memóriaigény-többletet jelenthet. Márpedig majdnem minden sablonhoz rendelt értékre igaz, hogy azt egy helyen egyszer fogjuk felhasználni, onnantól fogva pedig nincs rá szükségünk. Jobb lenne tán csak a PHP-ból jól ismert változóhivatkozást (nem ténylegesen új változó, csak adott változó memóriaterületére mutató álnév) alkalmazni. Minden eldobható adatnál érdemes ezt az értékátadási módot megfontolni. Ha mellette döntünk, csupán az assign() eljárást kell assign_by_ref() testvérére lecserélni, és máris megtakarítottunk egy kis memóriát!

A {section} és {foreach} listákban nem feltétlenül kell a táblázatsorok háttérszínét a PHP-től kérni, ha annak sorszíne váltakozó. Bár ez megoldható , ha az átküldött tömbben a sorok színét is tároltatjuk, de elegánsabb megoldás a sablonban jelezni (és igény esetén módosítani), hogy váltakozó sorszínt szeretnénk. Ehhez a {cycle} címkét kell alkalmazni:

```
{section name=i loop=$nevek}
<tr class="{cycle values="sotet,vilagos"}">
  <td>{$nevek[i]}</td>
</tr>
{/section}
```

Kimenő elektronikus levelek megformálására is alkalmazható ez a sablonrendszer. Ehhez a Smarty objektum fetch() eljárása ad segítséget. Használata megegyezik a display()-ével, de az eredmény megjelenítése helyett azt visszaadja. Ezt a visszakapott értéket azután a mail() PHP-függvényünk rendelkezésére bocsáthatjuk.
Alapesetben a template és *template_c* könyvtárakat a Smarty ott keresi, ahol a hívó-PHP található. Az alapbeállítások tehát feltételezik, hogy ezek a könyvtárak nyilvánosan is elérhető területen vannak. Ez nem feltétlenül kívánatos tulajdonság. Szerencsére mint minden, ez is átállítható a Smarty objektum $template_dir és $compile_dir változóinak felülírásával. Egyiküknek sem kell tehát nyilvános területen lennie. Amennyiben nem teljes a bizalmunk a sablonokat szerkesztő személyekkel szemben, érdemes a Smarty objektum $security változóját true állapotba billenteni. Ha ezt nem tesszük meg, a sablonba mindenféle PHP-kód is beilleszthető lesz a {php} {/php} címkék között. Nem hiszem, hogy taglalnom kellene, miért is veszélyes ez. Emellett ilyenkor csak az általunk meghatározott könyvtárakból tudnak {include}-dal sablonállományokat beilleszteni. Hogy melyek ezek a könyvtárak, a $secure_dir Smarty objektumváltozóban adható meg. Az említett változó egy olyan tömb, amelyben akárhány ilyen könyvtárat felsorolhatunk.

Végszó

Véleményem szerint a Smarty megismerése a PHP-ben járatosak számára nem megterhelő feladat. A nyelvezet alapjait egy hagyományos weblaptervezéssel foglalkozó munkatársnak sem nehéz elsajátítania. A sablonelvű tervezés által több időnk marad a program jó megtervezésére, és mivel leíráskészítésre is rákényszerülünk (a sablonban mit min keresztül adunk át), a rendszerünk is átgondoltabb lesz.

Heiglig (Cece) Szabolcs (cece@phphost.hu)
Veszprémben él. Hobbija, kedvenc időtöltése és munkája is a programozás. Szabadidejében hajlamos kerékpárra pattanni, vagy baráti társaságban szerepjátékokkal foglalkozni.

KAPCSOLÓDÓ CÍMEK

A Smarty hivatalos oldala
➲ http://smarty.php.net
Szabó Dénes Smarty témájú előadásának anyaga
➲ http://www.phpconf.hu/slides/smarty/

Tartalomkezelés

Szereljük fel honlapunkat hírújságszintű tartalomkezeléssel – nyílt forrású program használatával.

Emlékszünk még a Web régi szép napjaira, amikor a webmester volt a mindenes, egymaga csinált mindent a grafikus tervezéstől kezdve az adatbázis-programozáson át egészen a DNS-táblamódosításokig? Ahogy a Web fejlődött, az egyemberes weblapok egyre ritkultak. Igaz ugyan, hogy még ma is viszonylag egyszerű dolog egy szál magunkban egy dísztelen honlapot létrehoznunk és karbantartanunk, de általában már a legkisebb szervezetek is megosztják a felelősséget a programozók, a tervezők és a tartalmat adó emberek között. Továbbá számos szervezet kifejezetten azt szeretné, ha a különböző tartalomtípusokért különböző emberek felelnének, miközben valamennyien teljes jogkörrel rendelkeznének egy adott rész felett.

Természetesen a kiadók világában ez nem számít újdonságnak. Még a gimnáziumi hírújságot szerkesztettem, de már akkor használtuk az Atex nevű szerkesztő- és szedőrendszert. Az Atexet több dologért is kedvelték, de leginkább azért, mert épp úgy működött, mint az újság. Az Atexet használó riporterek egy súlyos, egyedi billentyűzeten a ***küld*** gomb megnyomásával elküldték írásaikat a szerkesztőknek. A szerkesztők végignézhették, hogy milyen cikkek várakoznak elolvasásra, szerkeszthették is őket, sőt a cikket vissza is küldhették írójának, vagy továbbíthatták a szövegszedő részleghez. A felépítésből fakadóan tilos volt olyan cikkeket megnézni, módosítani vagy fogadni, amiket a feldolgozási folyamat következő résztvevőjének továbbküldtek. A „Torkig vagyok a sürgetéssel!" kiáltással felpattanó riporter képe romantikus és lelkesítő lehet, de valószerűtlen a mai világban, ahol a híripar igen szűk határidőkkel dolgozik.

Ahogy a weblapok elérik az újságok szintjét, nem szabad meglepődnünk azon, hogy olyan programokat kezdenek használni, amelyeket tartalomkezelő rendszereknek (Content Management System, azaz CMS) nevezünk, és ezek igen hasonlóan működnek a korábbi Atexhez. A dokumentumok szervezése, az emberek és a munkafolyamat irányítása azonban összetett feladat, különösen, ha minden ember igényeit egyetlen programcsomagba szeretnénk belepakolni. Így aztán annak ellenére, hogy a tartalomkezelés egyre több weblap számára létfontosságú, a CMS-ügynökök híresek arról, hogy feltupírozott, drága programokat adnak el, amelyek sokat ígérnek ugyan, de vajmi keveset nyújtanak.

Mit csinál a CMS?

A tartalomkezelés egyik nehézsége, hogy minden weblap igényei mások. Ezért aztán az üzleti CMS-programokat általában két részletben árusítják. A vásárló először az alapprogramért fizet, majd legalább még egyszer ugyanennyit kell fordítania a tanácsadó és támogatási szolgáltatásokra. Így a CMS-program nemcsak drága, de igen jelentős megvalósítási és kipróbálási időt is igényel. Más szavakkal a CMS általában közelebb áll az eszközkészlet fogalmához, mint a kész alkalmazáséhoz. A legtöbb ilyen eszközkészlet a következő elemeket tartalmazza:

- ***Felhasználók:*** ha a weblapon mindenkinek különböző jogokat szeretnénk biztosítani, nyilván minden felhasználónak külön bejelentkezőnevet kell adnunk. A CMS ennek megfelelően tartalmaz egy felhasználókezelő programot, ami elkészíti, törli, szerkeszti és letiltja a rendszer felhasználóit.
- ***Engedélyek:*** miként Linux alatt képesek vagyunk írási, olvasási és végrehajtási jogosultságokat kiosztani a különféle fájlokra vonatkozóan, általában a CMS-program is lehetővé teszi, hogy az oldal karbantartója különféle jogokat adjon a rendszer összes felhasználójának. Hírújsághasonlatunkhoz visszatérve: a riporterek bevihetik a tartalmat, a szerkesztők módosíthatják (vagy visszaküldhetik a riporternek), végül a kiadók nyilvánosan elérhetővé tehetik (vagy visszaküldhetik a szerkesztőnek).
- ***Csoportok:*** bár elméletileg az egyes felhasználókhoz is rendelhetünk jogokat, ez a módszer hamar fárasztóvá válhat. Ezért a legtöbb CMS-program lehetővé teszi, hogy a felhasználókat az engedélyek kiosztása érdekében csoportosítsuk. Például megadhatjuk, hogy Tom, Dick és Harry írhatnak, szerkeszthetnek, de nem publikálhatnak, vagy ezeket a jogosultságokat hozzárendelhetjük az ***általános nevek*** csoporthoz – a hatás ugyanaz lesz.
- ***Sablonok:*** számos sablonozórendszer létezik, ilyen a JSP, HTML::Mason és a PHP is. A legjobbak szétválasztják egymástól a tervezést és a programlogikát, így a tervezők, írók és programozók anélkül tudnak egyszerre dolgozni a lapon, hogy egymás tyúkszemére lépnének.
- ***Kiadás:*** a Web kétélű fegyvere az azonnaliság. Amint módosítjuk kiszolgálónkon a ***foo.html*** lapot, mindenki láthatja, min változtattunk. Mi történik, ha hibázunk? Mi lesz, ha előbb ki szeretnénk próbálni? A CMS megoldása erre az, hogy minden tartalomdarabot kiadottnak kell megjelölni, mielőtt az láthatóvá válna a nagyközönség számára. Amíg a cikk nincs kiadva, addig láthatatlan.
- ***Előzetes és előnézet:*** miképpen a hírújságok és magazinok kiadói is szeretnék látni, hogyan fog kinézni a végleges termék, mielőtt elkezdenék nyomtatni a tényleges másolatokat, azonképpen a webkiadók is meg szeretnék tekinteni az oldalt, mielőtt az élővé válna a Weben. Ezért sok oldal futtat előzetes nézetet tartalmazó kiszolgálót, ami a legtöbb szempontból azonos a kiadási kiszolgálójukkal, attól eltekintve, hogy a világ elől el van rejtve. A kipróbálás ezeken az előnézeti kiszolgálókon folyik; amikor a szerkesztő vagy a kiadó elégedett, a tartalom átkerül a kiadási kiszolgálóra. Egy igazi CMS szinte bizonyosan lehetővé teszi, hogy ilyen formában állítsuk fel a rendszerünket.
- ***Munkafolyamat:*** az előnézet készítése már a végső lépés, amit igen hosszú utazás előz meg a szerző munkaállomásától a kiadási kiszolgálóig. Azt a módot, ahogyan a tartalom végigmegy a rendszeren, munkafolyamatnak (work flow) nevezzük. A CMS feladata legnagyobbrészt abból áll, hogy

ezt a munkafolyamatot meghatározza és kezelje. A riporterek visszaránthassák a történeteket a szerkesztőiktől? Hány szintű szerkesztői rendszert szeretnénk? Hová kerülnek a tervezők? Ki adja ki a végső kiküldési utasítást a tartalomra? Az összes ilyen kérdést a CMS munkafolyamat része kezeli.

- ***Kiadási dátumok:*** a Web előnye, hogy a kiadott anyagok azonnal megjelennek. De mi a helyzet, ha a vállalatunk részvénymanipulációt szeretne bejelenteni, de ezt az adatot nem teheti közzé hétfő kilenc óra előtt? Ülhetünk a számítógép mellett, várva, hogy a mutató elérje a kilenc órát, majd lenyomhatjuk az ENTER billentyűt, végre mindenki előtt felfedve a dokumentumot. Avagy használhatunk CMS-t, ami általában lehetővé teszi, hogy megadjuk, mikor szeretnénk megjeleníteni a cikket, illetve mikor fog lejárni.
- ***Webalapú szerkesztés:*** bár komoly szövegszerkesztésre a webböngésző az egyik legrosszabb választás, a legtöbb CMS-rendszer lehetővé teszi, hogy bizonyos vagy az összes dokumentumot böngészővel szerkesszük. Igazság szerint szinte minden CMS lehetővé teszi azt is, hogy fájlokat töltsünk fel a helyi számítógépünkről. A webalapú szerkesztés olyankor lehet hasznos, amikor sietünk vagy éppen csak egy-két dolgot szeretnénk megváltoztatni. Természetesen amelyik CMS lehetővé tesz ilyesfajta szerkesztést, az azt is ellenőrzi, hogy a lapot szerkeszteni próbáló személy rendelkezik-e a megfelelő jogosultságokkal.
- ***Keresés:*** a legtöbb CMS-csomag valamilyen keresési lehetőséget is nyújt, így dokumentumainkat megtalálhatjuk a rendszeren.

Bár fenti listánk nem teljes, viszonylag átfogó képet ad arról, hogy milyen nehézségeket próbál a CMS megoldani. Mindenki sejtheti, hogy minden CMS egy kicsit más képességkészletet tartalmaz és más módszert alkalmaz e nehézségek legyűrésére. Minthogy a CMS az ideje nagy részét tárolással, lekéréssel és tartalomkövetéssel tölti, senkit sem érhet meglepetésként, hogy a CMS számára az adatbázis majdhogynem létfontosságú.
Az üzleti CMS-csomagok általában azt feltételezik, hogy üzleti adatbázisrendszert használunk, például Oracle-t vagy a Microsoft SQL Serverét. Ahogy az várható, a nyílt forráskódú CMS-programok általában úgy készülnek, hogy a legjobban az olyan nyílt forrású adatbáziskezelőkkel működjenek, mint a MySQL vagy a PostgreSQL. A Zope tartalomkezelő keretrendszere (Content Management Framework, avagy CMF), ami tulajdonképpen egy saját CMS készítéséhez szánt eszközkészlet, szintén adatbázist használ, de ebben az esetben az adatbázis külső relációs adatbázis helyett a Zope beépített objektum-adatbázisa (ZODB) lesz.

Tartalomkezelés kontra alkalmazásfejlesztés

Ha valaha már fejlesztettünk komoly webalkalmazásokat, valószínűleg azonnal feltűnik a CMS által felajánlott szolgáltatások és web-alkalmazáskiszolgálótól elvárt képességek közötti hatalmas átfedés. A legtöbb CMS-program egy web-alkalmazáskiszolgáló felett helyezkedik el, s annak lehetőségeit használja fel a HTTP-kapcsolatok, a felhasználók, a csoportok, a jogok és az adatbázis-API kezeléséhez. Tulajdonképpen a CMS volt a weben fejlesztett első népszerű alkalmazásosztály, mint ahogyan a szövegszerkesztők voltak az első alkalmazások a személyi számítógépeken.
Általában véve nem rossz dolog, hogy a CMS-program az alkalmazáskiszolgálóra épül, és különösen igaz ez a nyílt forráskód világában. Így ugyanis új modulokat adhatunk a CMS-maghoz, új dokumentumtípusokat kezelhetünk, megváltoztathatjuk a sablonokat, kibővíthetjük az adatbázist, új jogtípusokat és munkafolyamat-szabályokat adhatunk meg. Nem szabad azonban megfeledkezni az alkalmazáskiszolgáló és CMS közötti különbségekről sem. Az első hátteret biztosít az alkalmazások kifejlesztéséhez, a második egy olyan alkalmazás, amit testreszabhatunk.
Amennyiben webalapú újságot, magazint vagy vállalati hírlapot szeretnénk létrehozni, nem kétséges, a CMS az, amire szükségünk van. Ellenben ha olyan webalapú alkalmazást szeretnénk, ami kedvenc jótékonysági intézményünk adományait kezeli, könnyen lehet, hogy a CMS-ben nem találjuk meg a szükséges rugalmasságot. A webalkalmazás és a webkiadás közötti különbség mindig is egy kicsit homályos volt, de ahogy a webalkalmazások kifinomultabbakká válnak, egyre inkább látható, hogy a CMS-program egy termék, amit webes felületen futtathatunk.
Mivel tartalomkezelőnk általában valamilyen alkalmazáskiszolgáló felett fog futni, a kiválasztott CMS nagyban függhet az általunk használt kiszolgálótípustól. Számos cég váltott át J2EE (Java 2 Enterprise Edition) alapra. Tulajdonképpen a jól ismert Vignette CMS-t is eredetileg Tcl nyelv alapúra tervezték, majd később átvitték J2EE alá, amikor a J2EE körüli láz olyan nagyra nőtt, hogy már nem lehetett figyelmen kívül hagyni. Mivel J2EE inkább szabvány, semmint termék, a vásárlók külön kiválaszthatják az alkalmazáskiszolgálójukat és CMS-programukat. Használhatjuk a Tomcat-JBoss párost vagy a BEA, az IBM és hasonló cégek üzleti ajánlatait.
Amennyiben nem kedveljük a Javát, vagy a fejlesztőcsapatunk más technológiákban jártasabb, kereshetünk nem J2EE-s CMS-t is. Ilyen termékek is léteznek, néhányukat meg is nézzük a következő hónapok során. Ilyen rendszer a Zope CMF-e, a CMF alapú Plone, Bricolage (Perl-PostgreSQL), PHPNuke-PostNuke/Xoops (PHP) és a Midgard (PHP).
A weblapkiadás területén a CMS használata egyre inkább szükségszerű lesz. Még akkor is érdemes lehet CMS-re váltanunk, ha csak egymagunk dolgozunk a weblapunkon, hiszen a honlapunkon segít szabványosítani a kinézetet és a tartalomtovábbítást.

Linux Journal 2003. április, 108. szám

Reuven M. Lerner (➲ http:/www.lerner.co.il/atf) Nyílt forrású programokra, valamint web- és adatbázis-alkalmazásokra szakosodott tanácsadó. Könyve, a Core Perl, 2002 januárjában jelent meg a Prentice Hall gondozásában. Reuven feleségével, és lányával Izraelben, Modi'in-ben él.

KAPCSOLÓDÓ CÍMEK

Az elfogulatlan CMS-adatok legjobb forrása a CMS-levelezőlista ➲ http://www.cms-list.org. A lista jelenlegi archívuma e cikk írásakor nem érhető el – de mint mondtam, a lista felbecsülhetetlen értékű áttekintést ad a CMS-piacról, üzleti és nyílt forrású termékek terén egyaránt.

A CMSWatch weblap (➲ http://www.cmswatch.com) vegyesen tartalmaz részletes jelentéseket, ingyenes tanácsadást, összefoglalókat és bemutató írásokat. Hasznos oldal, ha CMS-programokkal kapcsolatos anyagokra vadászunk.

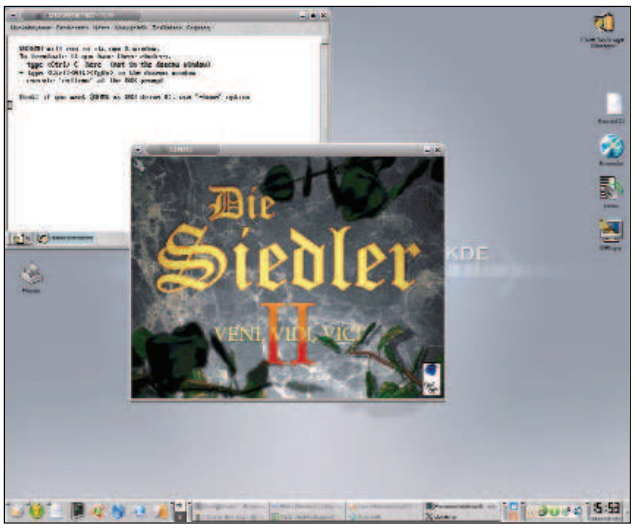
SuSE Linux 8.2 Professional

Aki már használta a SuSE termékeit, az tudja, hogy a cég alapelvei között szerepel a Linux nagyközönség számára is fogyaszthatóvá tétele. Ennek legújabb gyümölcse a SuSE Linux 8.2 Professional.

Elég régen használom a SuSE Linuxot, és eddig nem nagyon panaszkodtam rá. Igaz, most néhány Linux-guru biztosan a fejéhez kap, hogy miért nem használok „komoly Linuxot". Nem tudom. Nekem tetszik a SuSE hozzáállása az átlagfelhasználókhoz. Egyszerű, könnyen telepíthető programot adtak ki, és ez már az én legelső SuSE Linuxomra is igaz volt. De régen is volt… akkor még úgy hívták, hogy SuSE 6.4.

Kék korszak

Térjünk át a legújabb SuSE-termékre, a SuSE Linux 8.2-re. Gyorsan fel is tettem a gépemre, kíváncsi voltam, hogy mi változott az elődökhöz képest. Már induláskor látni, hogy valamiféle átalakulás történt, a szokásos üres képernyő helyett



1. kép A SuSE Linux 6.4 grafikus felülete

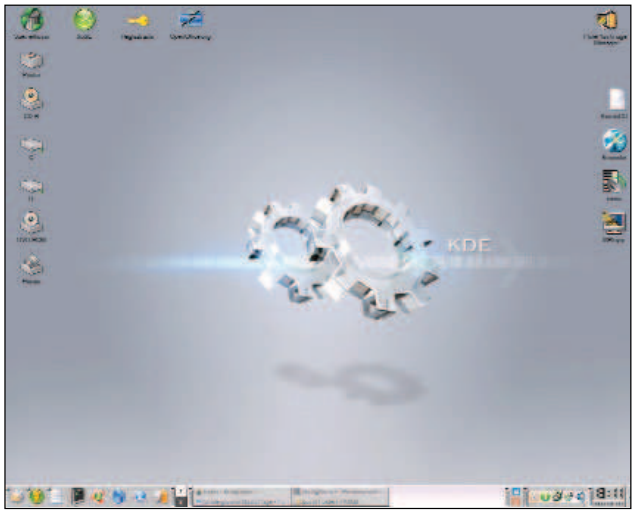
most egy szürke színű kaméleon jelenik meg kék háttér előtt, és körülötte sok üdvözlő szöveg látható. Ami viszont szemet szúr, hogy nincs közöttük magyar nyelvű üdvözlés (akárcsak az elődjénél). Igazság szerint nem is nagyon hiányzik onnan, de ha kínai írásjelekkel is üdvözölnek, talán a magyar is elférne. A SuSE nem hagy sokáig elmélkedni ezen a létkérdésen, megjelenik az indítómenü, amiben az újdonság a ***Boot from Harddisk***. Hasonló menüpont létezik a 8.1-es indítómenüjében is, de ott még ***Boot Installed OS*** néven szerepelt. Én most a telepítés (***Installation***) menüt választottam. Miközben a telepítéshez szükséges minimális rendszert tölti be, meg lehet csodálni az átdolgozott színeket: minden kék és a kéknek valamilyen árnyalata. Kis töltögetés után megjelenik a már ismert Yast2. Azonnal szemet szúrt a betűk élsimítása és a gombok „gömbölydedsége". A 8.1-eshez képest nagyságrenddel szebb a betűk kidolgozása. A következő telepítési pont a ***Telepítési beállítások***. Kellemes meglepetés ért, tudom használni egerem görgető szolgáltatását, ellentétben a 8.1-essel, amelyben csak a rendszer telepítése után sikerült. A következő változást a ***Rendszerindítás*** menüpontban találtam. Eddig ez a pont csak annyit tartalmazott, hogy a rendszer milyen, illetve ha több meghajtó is van, melyik merevlemezre lesz telepítve, most azonban két új bejegyzés került mellé. Az egyik, hogy milyen a rendszerbetöltőm típusa, a másik, hogy a telepítés után milyen bejegyzések kerüljenek az indítómenübe. Itt jegyezném meg, hogy az egyik legnagyobb újítás a 8.1-eshez képest az, hogy ez a rendszer már át tudja méretezni az NTFS fájlrendszert, így a program akár WinNT, akár WinXP mellé is telepíthető. Viszont észrevettem egy apró hibát is: akárcsak az elődje, ez a program is valami miatt időzavarban szenved. A program szerint most éppen 23:20 van. Ez barokkos túlzás, ugyanis nagyjából fél napot (12 órácskányit) siet. Miután mindent beállítottunk, elindulhat a telepítés.

A program telepítése

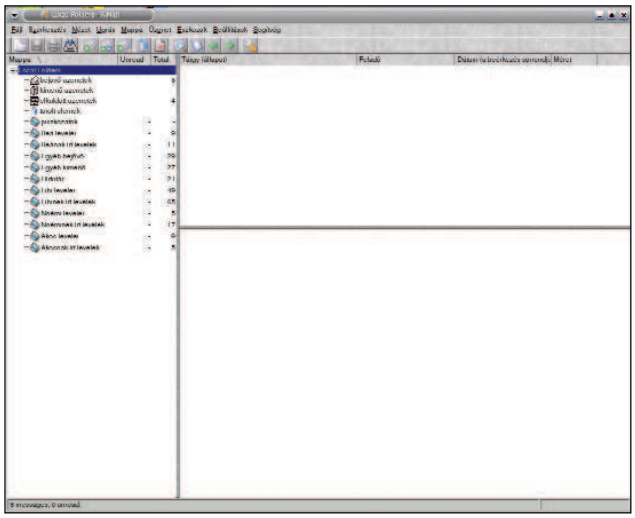
A telepítéshez az ***Elfogadás*** gombra kattintva megjelenik a szokásos zöld ablak, és megkezdődik a telepítés folyamata. Kicsit hosszadalmas lesz, mivel öt CD-t fog feltelepíteni, közel 1,5 GB terjedelemben. A 8.1-eshez még egy DVD-t is adtak, viszont a 8.2-höz már két DVD jár. Itt azonban szintén meglepetés érheti a felhasználót, ugyanis a telepítés során a képernyő alsó harmadában megjelenő adatok olvashatók. A 8.1-es ennél a résznél olvashatatlan adattömeget tartalmazott. Az első CD feltelepítése után a gép újraindul. Ennél a pontnál szintén felfedeztem egy kis változást: nem a szokásos karakteres felület jelenik meg, hanem egy kék háttéren lévő szürke kaméleon, de az F2-es billentyű lenyomása után visszakaphatjuk a karakteres felületet. Miután betöltötte magát, kéri a következő lemezt.

Beállítások

A telepítés befejezte után megjelenik a ***Hálózat beállítása*** menü, amiben a ***Hálózati interfészek***, ***DSL kapcsolat***, ***ISDN adapterek***, ***Modemek*** és a ***Proxy*** látható. Ebben a menüben a második és az utolsó menüpont új, amúgy minden a szokásos látványt nyújtja. Miután továbbléptünk, a következő ablak – ami szintén új az elődjéhez képest – az ***Internet-kapcsolat letesztelése***. Ha beállítottuk az elérést, akkor itt ellenőrizhetjük, hogy működik-e vagy sem. De ha valakinek nincs kedve ezzel bajlódni, egy kattintással át is ugorhatja.
A következő menüpont a ***Felhasználó azonosítási mód***. Itt állíthatjuk be, hogy egyedüli-e a gép, esetleg hálózatban van.
A 8.1-ben ezt a módot is csak a már teljesen beállított rendszerben a Yast lefuttatásával tudtuk beállítani. A következő menü egy felhasználó létrehozása. Itt is akadt egy kis gubanc: az első sor, amit ki kell tölteni, a ***Családnév***, utána a következő elméletileg az utónév lenne, de valamilyen érthetetlen oknál fogva itt is ***Családi név*** szerepel. Ha valaki tudja, mi a kettő között



2. kép A KDE 3.1.1.



3. kép
A Kmail 1.5.1, a fazonigazítás után már látható a könyvtárakban tárolt levelek száma

a különbség, akkor írja meg a címemre. Mindenesetre én a második ***Családi névhez*** a keresztnevemet írtam. Ami szintén új, hogy a gép figyelmeztet, ha a jelszavunk netán nem lenne elég bonyolult, például nincs benne szám vagy csupa kisbetűt írtunk. A ***Hardverbeállítások*** menüben alapvetően nem találtam változást, viszont a videokártya-beállítások kapcsán érdemes megemlíteni, hogy az nVidia lapkával szerelt videokártyák tulajdonosainak rossz hírrel kell szolgálnom: sem az `nvidia_glx`, sem az `nvidia_kernel` RPM-ek nincsenek benne, de még a `switch3nvidia_glx`-et sem leltem sehol. Mindent az nVidia oldaláról kell letölteni. Miután ezt a menüt is kivégeztük, a telepítés lényegében befejeződött.

KDE 3.1.1.

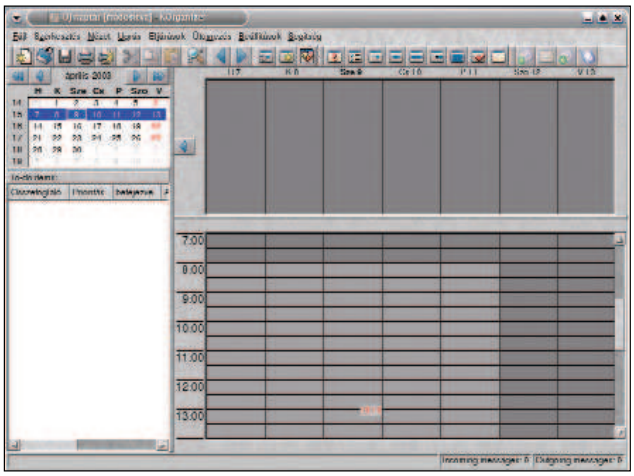
Kis töltögetés után megjelenik a grafikus felület bejelentkező ablaka. A háttér itt is kék, és a szokásos idegen nyelvű üdvözletek (magyar továbbra sincs) fogadnak. Nosza, jelentkezzünk is be! Rögtön az új KDE 3.1.1.-ben találjuk magunkat. A SuSE 8.1-es még csak a 3.0.1-et használta, ráadásul egy biztonsági rés is volt benne – igaz, a netről letölthető a KDE 3.0.4, amiben már kijavították a hibát. Ránézésre nincs különbség a KDE 3.0.4 és a 3.1.1 között. De ez csak egy felszínes megállapítás, ha ugyanis jobban szemügyre vesszük, észrevehetően szebbek az ikonok, valamint a szövegek is nagyon éles, kontrasztos hatást nyújtanak.

Amikor előhívtam a KDE menüjét, újabb meglepetés ért: a ***Rendszer*** almenüből is elindítható a Yast2! Az elődjénél ezt nem lehetett megtenni. Bár nem tudom, mi értelme volt ennek a lépésnek, ugyanis a Yast2 a régi helyén is megmaradt. A ***Vezérlőközpont*** szintén keresztülment némi fazonigazításon: szebbek lettek az ikonok és jobban áttekinthető a menürendszer. Amit viszont sajnálok, az az, hogy nincs képernyővédő. Pontosabban képernyővédő akad, de csak két választásunk lehet: az üres képernyő és a véletlenszerű. Ez az utóbbi igen érdekes lehet, amikor a SuSE véletlenszerűen próbálja meg elindítani azt az egyetlen képernyőkímélőt…

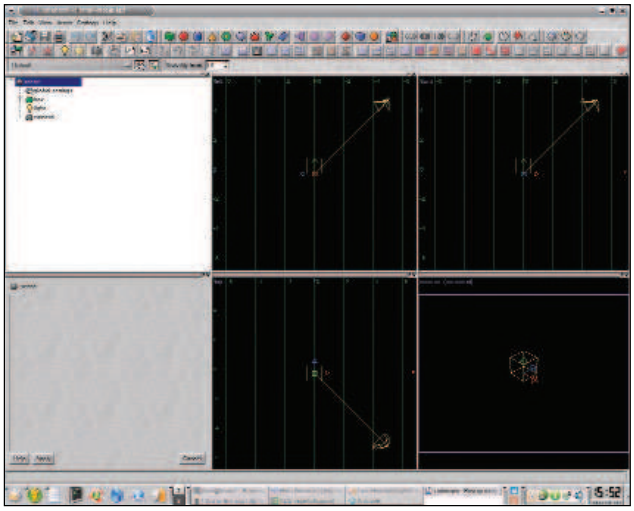
A ***Stílus*** és a ***Színek*** menüpont most már nem a ***Kinézet*** menü alatt található, hanem külön-külön önálló menüt alkotnak. Továbbá a Midnight Commander – akárcsak az elődjénél – itt sem települ önműködően, amit nem értek, lévén sok Linux-felhasználó szívesebben részesíti előnyben a karakteres fájlkezelőket, mint a grafikus felületűeket. A Yast2 ***Hálózati beállítások*** menüjének ***Modembeállítás*** almenüje is tartalmaz érdekességeket. Ha még nincs modem beállítva, akkor teljesen érthető, hogy leellenőrzi, van-e. Az viszont meglehetősen boszszantó, hogy akárhányszor a ***Modem*** menüt elindítom, mindig leellenőrzi ugyanezt, függetlenül attól, hogy már be lett-e állítva. Ez a belső modemmel rendelkező felhasználókat nem is nagyon érinti, inkább a külső modemesek fognak neki örülni, ugyanis időbe telik, amíg felismeri a külső eszközt. Továbbá azt is örökölte elődjétől, hogyha be szeretnéd állítani a modemet, az internetelérással is egyből meg kell tenned ugyanezt. Olyan gomb, hogy ***Mégsem***, sajnos nem létezik. Olyan létezik, hogy ***Elvet***, de akkor az egész modembeállítás megy a kukába. Esetleg a ***Következő*** gombra kattintva beállíthatom a szolgáltatómat. Ha már a szolgáltatóknál járunk, megemlíteném, hogy szinte az összes magyarországi internetszolgáltató benne van az adatbázisában. Újabb kellemes meglepetésre akadhatnak azok a felhasználók, akik Intracom ISDN adapterrel rendelkeznek. A 8.1-es alatt igencsak érdekesen lehetett beállítani a modemet. Most már ez is könnyedén megy. A 8.1-esben hiába állítottunk be mindent rendesen, nem működött, viszont a SuSE oldalán megtalálhattuk az útmutatást, hogy mit hogyan írjunk át a modem munkára bírásához. Most a Yast2 beállításai is elegendők ahhoz, hogy a modem működjön.

Ha már az internetelérésnél járok érdemes említést tenni a Konquerorról. Ez az a program, ami az elődjéhez képest elég

	SuSE 8.1 Professional	SuSE 8.2 Professional
Rendszermag	2. 4. 19	2. 4. 20
XFree86	4. 0	4. 3
glibc	2. 2. 5	2. 3. 2
gcc	3. 2	3.3
KDE	3.0.1	3. 1. 1
Gnome	2. 0	2. 2
FAT32-átméretezés	igen	igen
NTFS-átméretezés	nem	igen



4. kép
KOrganizer, ami már képes lekérdezni a Microsoft Exchange2000 kiszolgálókat



5. kép A népszerű Povray modellezőfelülete

komoly változáson ment keresztül. Talán az egyik leghasznosabb tulajdonága a tabok kezelése, ezzel lehetővé vált, hogy több ablakot is megnyissunk egy böngészőben. De akinek még ez sem elég, az ketté is oszthatja az aktuális ablakot és mindkét felében más-más oldalt jeleníthet meg, bár ehhez legalább 19”-os monitor szükséges, hogy élvezhető maradjon az oldalak mérete. Sajnos a tabok nem „osztódnak”, ezért, ha egy tabfülre kattintunk önműködően megszűnik a felezés és betöltődik a tab. Ami szintén új az a beépített oldalfordító. A program képes az adott oldal tartalmának a lefordítására. Nyolc nyelvet ismer (angol, német, francia, olasz, japán, kínai, portugál, spanyol), sajnos a magyar nincs közöttük.

X-újdonságok

Az is újdonságnak számít, hogy az X-felület felbontását az CTRL-ALT-SZÜRKEMÍNUSZ billentyűkkel átállíthatjuk. Az asztalon lévő ikonok, megnyitott programok önműködően átállnak az új felbontásra. A program változatszámán is lehet látni, hogy látványos, nagy változások nem igazán történtek, inkább csak apró csiszolgatások, abból viszont rengeteg. Lényeges újítás, hogy a 8.2-es SuSE már a 2.4.20-as rendszermagot használja. A rendszer közel 1200 meghajtót tartalmaz, ezenkívül a Xfree86 4.3-as a gcc3.3-as és a Sun Java 2 1.4.1-es változatát

6. kép
A Konqueror 3.1.1 ketté tudja osztani az ablakot és már a tabokat is kezeli

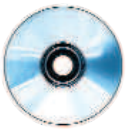
is magában foglalja. Továbbá tartalmazza az OpenOffice.org 1.0.2-es változatát, magyarul. Itt említeném meg, hogy az OpenOffice gyorsindítójának köszönhetően az irodai csomag indításakor keletkező várakozási idő már a múlté. Egy kis összehasonlító táblázatba összeszedtem a 8.1-es és 8.2-es főbb jellemzőit (lásd 75. oldal).

A piacon keringő számos Linux-változat közül talán a SuSE az, amelyik egy átlagos felhasználó igényeinek a legjobban megfelel. Igaz, hogy akadnak még gyerekbetegségei (lásd modembeállítás, időkezelés stb.), de még így is az egyik legkönnyebben beállítható Linux. Minden olyan felhasználónak javaslom a kipróbálását, akinek elege van Windows rendszere lefagyásaiból és érthetetlen hibáiból. A SuSE Linux felhasználóbarát rendszer, ami abból is látszik, hogy nem terheli az embert különböző kérdésekkel. Ha jól számoltam, akkor három, esetleg négy kérdést tesz föl, és a végeredmény egy működő Linux-rendszer, ami szinte bármilyen környezetben megállja a helyét, legyen az kedvtelés vagy irodai munka. Ez a rendszer egyszerű telepíthetőségével komoly vetélytársa lehet a másik ablakos rendszernek. Megbízhatóságában viszont már most messze előtte jár.

Végezetül ejtsünk pár szót a program anyagi vonatkozásairól. A végfelhasználói ár a 24 ezer forint. Ebben benne foglaltatik az öt CD és a két DVD, továbbá egy 1200 oldalas magyar nyelvű leírás, valamint három hónapos telepítési segítségnyújtás telefonon vagy elektronikus levélen keresztül. A program megjelenése 2003 áprilisának második felére várható (lapunk nyomdába adása utánra – a szerk.).

Lapunk lemezmellékleteként egy telepíthető, egykorongos változatot adunk. Természetesen ezen korán sincs rajta minden program, arra azonban mindenképpen jó, hogy kedvet kapjunk a teljes csomaghoz.

Molnár Tibor (tibor@hu.inter.net)
A GDF végzős hallgatója. Linuxszal közel négy éve, míg számítógépekkel 18 éve foglalkozik, s hobbijai is erre a területre összpontosulnak. Bár van egy fekete bárány is a kedvtelései között: szereti a ritka autókat.

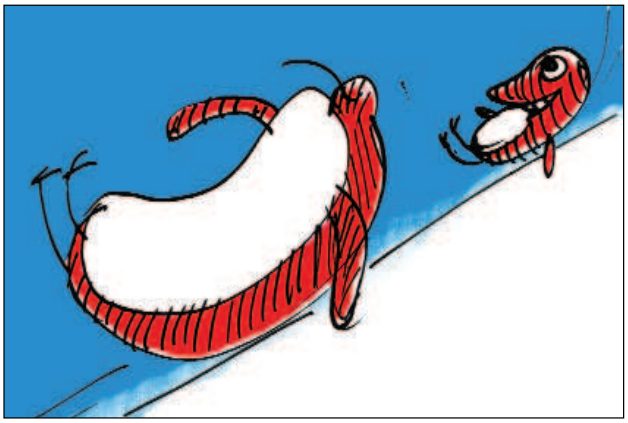


Játszótér

Végre eljutottunk idáig. A Linuxvilág hasábjain is bebizonyíthatom, hogy nem csak Windows operációs rendszeren léteznek játékok.

Nem arról a több tucat ingyenes programról beszélek, amelyek minden egyes terjesztésben benne vannak, sokkal inkább azokról, amelyeknek létezik linuxos változatuk vagy eleve csak Linuxon adták ki a fejlesztők. Nagyon sok időnek kellett eltelnie ahhoz, hogy találjak valakit, aki végre komolyan veszi a témát – mert valljuk be, ha egy Linux-kedvelő bemegy egy számítástechnikai boltba, akkor a Linux szó hallatán majdhogynem csak néznek, és nem értik, mit akar a kedves vásárló. Még inkább igaz ez a jelenség, ha valahol linuxos játékokat tartalmazó CD-t próbálsz meg venni. Számomra elfogadhatatlan volt és a mai napig is az, hogy a sok-sok windowsos és konzolos játékot tartalmazó CD mellett miért nincs ott a polcokon a linuxos változat is. Azért sem értem a dolgot, mert a „tábor" bővül, folyamatosan nő azoknak a száma, akik a Linuxot nemcsak fejlesztésre, webkiszolgáló futtatására, hanem már munkaállomásokon és a játékaik futtatására is használják. Sajnos gyakran tapasztaltam, hogy egy-egy felhasználó csupán a közszájon forgó tévhitek miatt hajtogatta: minek nekem Linux, főleg játszani, amikor az XY boltban vásárolt géphez „ingyen" telepítik a Windows valamilyen változatát, illetve kiegészítőként az Office programcsomagot, ami sokkal jobb és tökéletesebb, mint a Linux! Emlékszem, egyszer egy tévéadásban a betelefonáló elmondta a véleményét, és nem értettem, miért nem tud egyetlen épkézláb indokot sem felhozni arra nézvést, hogy miért találja jobbnak a Windowst. Minden fennhéjázás nélkül állíthatom, hogy a lassan két éve megjelent leírásaim manapság is felbukkannak a linuxos oldalakon, és mindenütt legalább egy-két sor erejéig foglakoznak a Linux „játékos" oldalával is.
Rovatunk lényege gyakorlatilag ugyanaz lenne, mint sok más játékrovaté, ami a sajtóban megjelenik: minden hónapban kerül bele hír, érdekesség, tesztek, demók, játékleírások, alkatrészpróbák és minden, ami kapcsolatba hozható a linuxos játékokkal. Szeretnék egy levelezőrészt is beiktatni a rovatba, amin keresztül segítséget tudnék, illetve tudunk nyújtani minden kezdő linuxos játékkedvelőnek. Jelenleg egy olyan CD összeállításán dolgozom, amin csupa linuxos játékkellék, frissítés, telepítőprogram stb. jelenne meg. Várom az igényeket, hogy ki mit szeretne „kapni" ezen a CD-n.
Cikksorozatom útra bocsátásakor néhány dolgot előre el kell mondanom. A gép kiépítése, amin a játékokat kipróbálom és a leírásokat készítem, a következőképpen fest:

- iiyama LCD monitoron jelennek meg a képek,
- egy HP Vectra VL420 DT gép, amiben
- 1,6 GHz-es a processzor,
- 320 MB RAM,
- 10 GB üres hely van a linuxos lemezrészen,
- nVidia GeForce 2 MX400 kártya (a Linuxvilág főszerkesztőjének jóvoltából),
- Intel810 AC97 hangvezérlő az alaplapra integrálva,
- Intel PRO/100 VE hálózati kártya, szintén az alaplapon.



A kipróbáláshoz Linux a Red Hat 8-at használtam, mivel a 9-es jelenleg még nem érhető el.
Felvetődik a kérdés, hogy miért pont a Red Hatet választottam? Törvényes és ingyen elérhető, nem úgy, mint a Xandros Desktop 1.0, amit a múlt havi számban mutattam be. Az RPM alapú rendszerek általában a kezdők számára alkalmasabbak a tanulásra, kezdésre.
Szándékomban áll egy „játékszótár" összeállítása, illetve a

linuxos játékok ismeretéhez szükséges elemző rész létrehozása, ami részletesebb magyarázatokat fűz azokhoz az alapfogalmakhoz, amelyeknek az ismerete elengedhetetlen egy szuperjátékos számára.

Hogy mire gondolok? Például a GLX, GLCore, GLUT, OpenGL, OpenAL, Alsa, DRI stb. olyan varázsszavak, amik legalább olyan érdekesek, fontosak, mint a Windows világában a DirectX.
Felvetődik még egy izgalmas kérdés: milyen játékokat próbálunk ki az elkövetkezőkben? Jelen pillanatban a Loki által készített majdnem teljes játékkínálat rendelkezésre áll, mintegy 95 százalékban, valamint az IdSoftware termékei, a ➲ http://www.tuxgames.com oldalon elérhető és megvásárolható játékok mintegy nyolcvan százaléka, valamint a Transgaming által fejlesztett WineX, ennek segítségével gyakorlatilag (bizonyos korlátok között) a windowsos játékpaletta is elérhetővé és futtathatóvá válik kedvenc Linux operációs rendszereden.

Dióhéjban a DRI-ről

A DIR (Direct Rendering Interface), közvetlen leképező adaptert jelent.
A DRI három részből áll:
1. valamilyen OpenGL-változatból (például Mesa ➲ http://www.mesa3d.org/)
2. GLX-ből ➲ http://utah-glx.sourceforge.net/ vagy ➲ http://www.sgi.com/software/opensource/glx/
3. Linux-közeli meghajtókból

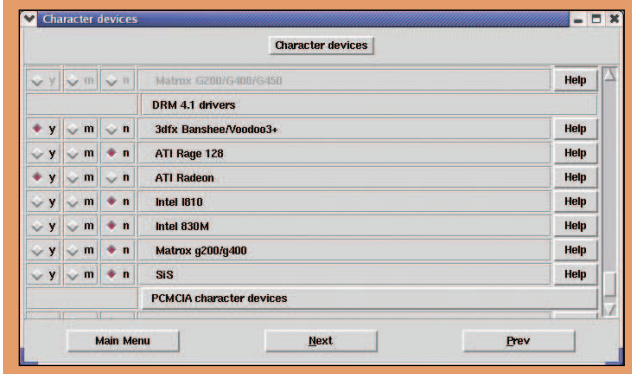
A DRI gyakorlatilag része az XFee86-nak, és ha az általad használt grafikus kártyának nincs a gyártó által készített meghajtója, akkor a rendszermagmodul és az Xfree86 próbálja meg a kártyából a grafikus gyorsítást kinyerni. Hátránya, hogy közvetlen leképezést végez, így az alkalmazás, tehát a 3D-s játék közvetlenül a kártyát éri el (gyenge kártya esetén – mondanom sem kell, hogy – lassabb, rosszabb a gyorsítás). A közvetlen kártyaeléréshez viszont az agpgart használata ajánlott a magban, mivel az agpgart és a drm-meghajtók, valamint az xfree86 és a grafikus kártya összessége adja a 3D-gyorsítást.
A DRI-t, mivel az XFree86 része, minden Linux-terjesztés tartalmazza, és az esetek többségében a telepítőprogram be is állítja, mint ahogy megtette Red Hat 8 esetén is.
Az X-oldali beállítások gyakorlatilag az alábbiakban merülnek ki: a */etc/X11/XF86Config* fájlban, a `Module` részben szerepelnie kell a

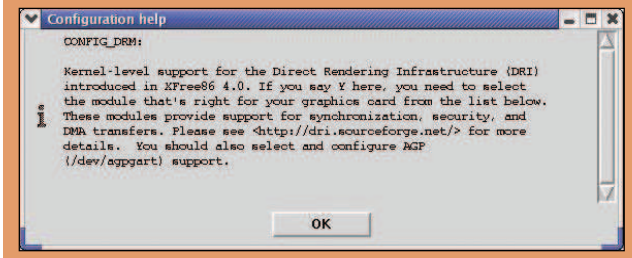
```
Load "dri"
Load "glx"
```

bejegyzésnek, valamint általában a fájl végén található a

```
Section "DRI"
      Mode 0666
EndSection
```

A megfelelő meghajtónak szerepelnie kell az XFree86 által támogatott kártyák között (➲ http://www.xfree86.org/current/Status.html). Ezenkívül ellenőrizni kell a már említett magoldali részt (agpgart, DRM-meghajtók), és ekkor már semmi nem ment meg attól, hogy használatba vegyük a DRI nyújtotta lehetőségeket.
Ellentmondásos viszont, hogy az X futása esetén az nVidia meghajtója kifejezetten rosszul viseli a `dri` modul jelenlétét, így – mint az a leírásból kiderül – a `dri`-re vonatkozó bejegyzéseket el kell távolítani az *XF86Config* fájlból. Minden más kártya esetén viszont célszerű a DRI használata, mert határozott gyorsulás érhető el vele a 3D-megjelenítés terén.
➲ http://dri.sourceforge.net/





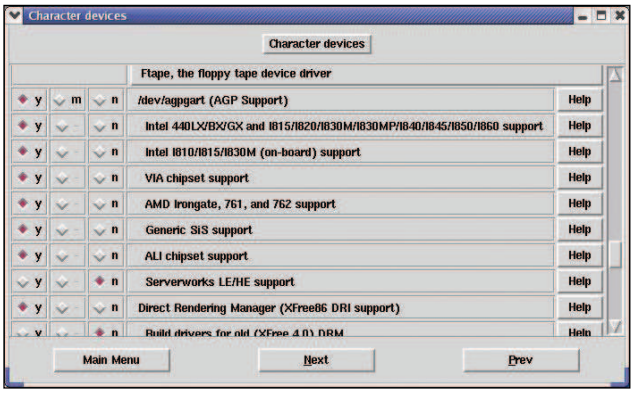
Újdonságok

A hónap híre számomra, hogy a Microsoft rátette a kezét a Bioware-re, és kérdésessé vált, hogy ezek után egyáltalán meg fog-e jelenni a Neverwinter Nights linuxos ügyfél (az nwn linuxos kiszolgálórésze már nagyon régen elkészült). Üdvözítő, hogy napok alatt kijött a próba1, 2, majd nem sokkal a cikk írása előtt a próba3-as változat is. Örömteli ez már csak azért is, mert a tavaly novemberi megjelenés eléggé csúszott. Mivel egyre gyorsabban jönnek ki a próbaváltozatok, remélhetőleg a végleges változatot is hamarosan ki tudjuk próbálni, s addigra jó lenne egy telepítő is, nem csak egy tarfájl.

A programról, illetve a telepítéséről bővebben akkor szólunk, ha a végleges ügyfélváltozat elkészül, ami az utóbbi idők gyors egymásutánban megjelenő próbakiadásait figyelembe véve akár már a jövő hónapban is lehetséges lesz (➲ http://nwn.bioware.com).
Szintén kiemelkedő jelentőségű, hogy megjelent a legújabb nVidia-meghajtóprogram is (➲ http://www.nvidia.com/view.asp?IO=linux_display_ia32_1.0-4349), amit sokan szidnak, ám az nVidia alapú kártyák esetén követendőnek számít. Érdekes telepítővel látták el (Loki-fejlesztés), ami az új jellemzőkkel kezelhetőbbé válik. Az nVidia állítása szerint támogatja a Red Hat 9-et is. Sajnos ezt csak azok tudják megerősíteni, akik előfizettek a Red Hat Network szolgáltatására és letöltötték a lemezlenyomatfájlokat.
Az nVidia szerint a telepítés négy lépésből áll:

1. felhasználási szerződés és az ***Olvass el*** fájl áttekintése;
2. a meghajtó letöltése;
3. a függőségek ellenőrzése;
4. a meghajtó telepítése.

A meghajtóprogram letöltését követően érdemes ellenőrizni, hogy futtatható-e. Az egyszerűbb telepítés gyakorlatilag a fájl futtatásából, illetve két ENTER lenyomásából és az ***XF86Config*** fájl szerkesztéséből áll. Az ***XF86Config*** fájl a */etc/X11* könyvtárban helyezkedik el. Mit is kell szerkeszteni benne?
1. A `Section "Module"` bejegyzést kell keresni, majd a Load „dri” sort vagy ki kell törölni, vagy a sor elejére # (kettős keresztet, azaz megjegyzésjelet) kell tenni.
2. Ha az indítandó modulok között szerepel a `Load "GLcore"`, akkor ugyanez az eljárás.



3. Amennyiben a `Load "glx"` sor nem szerepelne az indítandó modulok között, akkor fel kell venni.
4. A `Section "Device"`-t és a `Driver "nv"`-t le kell cserélni `nvidia`-ra.
5. A `Section "DRI"`-t, ha van ilyen bejegyzés, megjegyzésbe kell tenni vagy ki kell törölni.

Ez volt az egyszerűbb módszer, lássuk a bonyolultabb változatot! A fájlt a `-extract-only` kapcsolóval kell futtatni, ekkor a fájl nevével azonos könyvtár jön létre, ahol az nVidia telepítője, valamint a meghajtó forrása található. A könyvtárba lépést követően ki kell adni a `make install` parancsot, és az ***XF86Config*** fájlban ugyanazokat a beállításokat kell elvégezni, mint az egyszerűbb telepítés esetén.
Felmerülhet a kérdés, hogy miért is jó, ha nem a telepítőt használjuk. Azért, mert a forrás könyvtárszerkezetébe tekintve egy érdekes fájlra bukkanhatunk, ami a */NV…/usr/src/nv* könyvtárban rejtőzik. A fájl neve ***os-registry.c***, ebben nagyon sok kapcsolót lehet és érdemes átállítani.

A TNT-kártya beállítása

Például, ha valakinek régi TNT alapú kártyája van, meg kell keresni az `Option: VideoMemoryTypeOverride` bejegyzést a fájlban, és két kapcsolót kell benne megadni (a teljes bejegyzést lásd az 1. listán a 47. CD Magazin/Játék könyvtárában).

```
1=SDRAM van a kÆrtyÆn
2=SGRAM van a kÆrtyÆn
```

Régebben nekem is Hercules Dynamite TNT-m volt, és e kapcsoló nélkül nem működött a meghajtóprogram.

Az AGP megadása

Például azt is a ***os-registry.c***-ben kell beállítani, hogy milyen AGP-t szeretnél használni (lásd a 2. listán a 47. CD Magazin/Játék könyvtárában). 0 érték esetén az nVidia saját AGP-kódját engedélyezed, míg az 1-es értékkel letiltod azt, és a Linux-rendszermag által használt AGP-t fogja a meghajtó használni. Ugyanilyen eljárással lehet paraméterezni az AGP írási sebességét is (`AGPFW >> AGP Fast Writes`). A fájl szerkesztését követően ki kell adni a `make install` parancsot, és a rendszermagmodul az új jellemzőkkel fog lefordulni és települni.
Az ***XF86Config*** fájlban a `Screen` bejegyzést kell keresni, és az alábbiakat kell begépelni:

„NvAGP” „egy egész szám”
0 = nem használ AGP-t;
1 = nVidia belső AGP-t használ;
2 = a rendszermag által támogatott agpgart-ot használja (amennyiben a magba be van fordítva a támogatás);
3 = bármilyen elérhető AGP-t használ. Ha nem tudod, milyen érték van a magban, vagy nem tudod, hogy mit állítottál be az ***os-registry.c***-ben, akkor ezt érdemes használni (az alapértelmezett a 3-as).

Az AGP elérési sebességének beállítása

Ha például biztos akarsz benne lenni, hogy 4× AGP-t használ a meghajtó, akkor egyértelműen át kell írni a 7-est 4-re, mert így a meghajtó nem a szóba jöhető összes sebességet kéri le a rendszertől, a csupán 4×-est használja.

Az nVidia logó megjelenésének kikapcsolása

Ha nem szeretnéd minden X-induláskor az nVidia logót nézni, akkor ezt a kapcsolót kell elhelyezned az XF86Config fájlban:

```
Option "NoLogo" "boolean"
# Letiltja az nVidia log  megjelen tØsØt
# az X ind tÆsakor
# AlapbeÆll tÆs szerint a log  megjelenik.
```

Az `Option "NoLogo"` (logikai változó) szintén a Screen bejegyzésnél kell keresni és oda beírni.

A régi monitorok beállítása

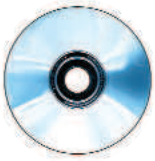
A következő paraméter a régi monitoroknál segít, ha például a meghajtó feltelepítése után a felbontáshoz képest nagyon nagyok a menük, betűk stb.

```
Option "NoDDC"
```

Az eddigieket összegezve megállapíthatjuk, hogy a ***os-registry.c***-ben érdemes a finomhangolást elvégezni.
A jövő hónapban a teljes Quake-kínálat lesz a porondon. Némi nehézség árán megpróbálom futásra bírni a Quake első változatát is. Természetesen nem marad ki a 2-es és a 3-as sorozat sem, kiegészülve a Team Arena CD-vel.

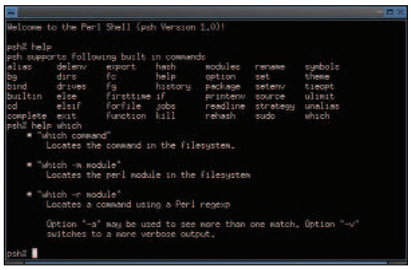
Kosztandinovszki Norbert
(kosztandinovszk@dialec.hu)
Linux- és játékmániás számítógépőrült.

Épkézláb parányok



psh

Íme, a Perl-bűvészek parancsértelmezője! Számos Bash-szolgáltatással rendelkezik, de a Perl szempontjából valamivel barátságosabb. Még csak nemrég kezdtem bejelentkezési héjként használni, így egyelőre nem mondhatok róla sokat. Mindenesetre határozottan könnyebben kezelhető. Ha kétségeid vannak, kérdezz meg egy év múlva, hogy még mindig ezt használom-e. Hiszem,

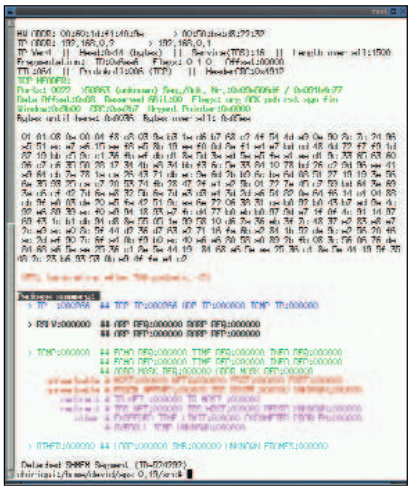


hogy eljön a nap, amikor a Perl átveheti az összes rendszerszegédprogram szerepét (ha valakinek ilyen hajlamai vannak), addig is megelégszem egy Perl-héjjal. Futtatásához Perl szükséges és BSD::Resources (választható).

➲ http://www.focusresearch.com/gregor/psh

Advanced Packet Sniffer

Mint azt már három évvel ezelőtt is megjegyeztem, amikor először vizsgáltam meg a fenti alkalmazást, ez a hálózati figyelőeszköz nem hasonlított a `tcpdump` programra. Segítségével láthatóvá válik a csomagforgalom, ami

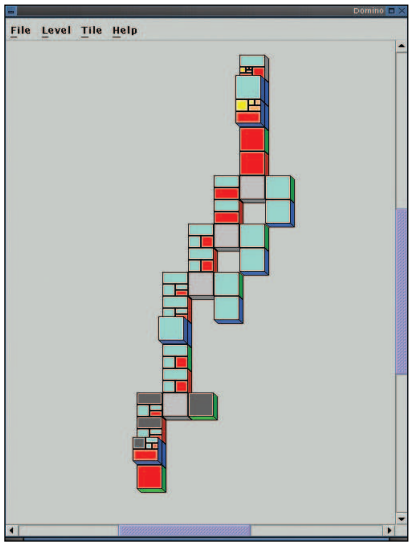


lehet nagyon érdekes is, vagy éppenséggel teljesen értelmetlen – különösen titkosított kapcsolatok esetén. Ez a program részben azért tartozik a kedvenceim közé, mert meg lehet vele mutatni, hogy milyen olyan adatok mennek a hálózaton, amit bárki szabadon elolvashat. Futtatásához glibc szükséges.

➲ http://www.swrtec.de/clinux

Domino on Acid

Különösen nehéz játék, azonban saját „tippek és trükkök” kézikönyvvel bír. A játékot letöltheted, vagy közvetlenül az Interneten is játszhatod. Saját gépeden jar-fájlként indíthatod, egyébként bármelyik Java-képes böngészőből meg



lehet hívni. Ha úgy játszol, mint én, nálad is mindent el fognak borítani a téglák, és semmivel sem jutsz közelebb a megoldáshoz. A futtatásához Java szükséges.

➲ http://www.winterdrache.de/freeware/domino/index.html

Checkbot

A Checkbot egy egyszerű, kis erőforrásigényű weblapellenőrző program. Ha a `--match` kapcsolóval, a saját tartományod nevét megadva futtatod, nem kalandozik el azokra az oldalakra, amelyekre esetleg hivatkozol. Az eredményt levélben is megkaphatod, vagy figyelemmel követheted, amint weboldaladat a talált címek alapján bejárja. Nincs több hibás hivatkozás, bármilyen összetett legyen is az oldal – vagy ha mégis, legalább tudsz róla. A futtatáshoz Perl, valamint a File::Basename és LWP Perl-modulok szükségesek.

➲ http://degraaff.org/checkbot

Dbmail

A Dbmail többezer felhasználós, nagy levelezőkiszolgálók esetén hasznos. A leveleket és a felhasználókat SQL-adatbázisban tárolja, így nem kell rendszerfelhasználókat létrehozni. A Dbmail program nemcsak a `dbmail-smtp`-vel használható, ami csak egy SMTP-fogadódémon (emellett szükség van a Sendmailre vagy más MTA-ra a levelek elküldéséhez), hanem a `dbmail-pop3d` POP3-kiszolgálódémonnal, illetve a `dbmail-imapd` IMAP-démonnal is. A mellékelt beállítási leírás igencsak vázlatos, úgyhogy magadnak kell kitalálnod – szerencsére nem túl bonyolult. Az SMTP előtt a POP-ot vagy az IMAP-ot is be kell állítani. Előfeltételei: PostgreSQL vagy MySQL, libssl, libcrypto, glibc.

➲ http://www.dbmail.org

Gtk-Perl Flash Cards

A villámkérdés jól bevált tudásfelmérési módszer. Kapsz egy kérdést, válaszolsz rá, és ellenőrzöd, hogy helyes-e a válasz. Ezen az elven alapul ez a kis alkalmazás. Megadhatod a kérdések nehézségi szintjét, tetszés szerinti számú kérdést vehetsz fel az adatfájlba, azután ellenőrizheted a tudásodat. A program számon tartja a helyes és helytelen válaszok számát, az összes kérdés számát, lehetővé teszi a véletlen lekérdezést és egyéb szolgáltatásokat is tartalmaz. Perl és a Gtk Perl-modul szükséges a futtatásához.

➲ http://www.masswire.com/flash.tgz

A programok megtalálhatóak a 47. CD Magazin/Paranyok könyvtárában.

Linux Journal 2003. április, 108. szám

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