Múlt hónapban írtam, hogy nagyon sok gondot okoz a szakkifejezések használata, egyrészt a kezdőknek, hiszen számukra az új területet még nehezebb megismerni, ha új, az élet más területeiről sem ismerős szavakat kell megtanulniuk, másrészt pedig a gyakorlott szakemberek számára is, hiszen aki például egy angol szakkönyvből tanulta a hálózatfelügyelet alapjait, hiába van tisztában a NAT vagy a masquerading fogalmával, vakargathatja a fejét, hogy mit is jelent a ***címfordítás*** vagy az ***álcázás***.

Örömmel fogadtam leveleiteket, véleményeiteket a szótárról, igyekszünk ezt az oldalt rendszeresíteni. Természetesen nehéz feladatról van szó: egy-egy kifejezéshez például mi magunk találtunk ki magyar megfelelőt, másokat hosszú-hosszú évek óta használja a szakma, ilyen például az ***adatsín*** kifejezés a databus helyett, vagy az ***ismétlő*** a repeater helyett.

Külön örültem, amikor a ***helyőrrel*** kapcsolatban kaptam véleményt. Igaz, nem az én „gyermekem” – először ***Borai János***-tól hallottam, és nagyon tetszik. Nem tudom, hogy húsz év múlva nyugodtan használják-e, vagy – ahogy a nyelvújítás alatt a rengeteg értékes szó mellett készült egy-két borzalmat szokták használni – elrettentő példaként lengetik majd a lurkók szeme előtt. Remélem, azért akad majd olyan szó is, ami szakmai és nyelvi szempontból is megállja a helyét, sőt még a tanárok is megkedvelik, mert bár gyakran lekicsinylik a tanárok szerepét e szempontból, éppen a számítástechnikával való ismerkedés időszaka hagy mély nyomot.

És továbbra is vannak a kemény diók. Ilyen például a ***globális–lokális*** páros. Annak ellenére, hogy a lokális elfogadott párja a helyi (akár hálózatról, akár változóról, vagy bármi másról legyen szó), a globálishoz már nem találunk olyan könnyen párt. Sokszor használják az átfogó, a teljes hatókörű, a mindenhol látható vagy éppen a közismert szavakat, ezek közül a változókra nehezen mondhatnánk akármelyiket is.

Vagy ott van az ***orientált*** szó, melynek jelentése irányult, központú, esetleg alapú lehet. Hosszú viták folytak már e szó körül, sokan más értelemben használják, például az objektumorientált külön gond, ugyanis, ahogy erre rávilágítottak vitapartnereim, önmagában is ellentmondásos szakszó. Sajnos, ez egy olyan kulimász, amiben nem lehet egy pillanat alatt rendet teremteni, de remélem, hogy előbb-utóbb letisztulnak a jelentések.

Lássuk tehát az e havi adagot, és továbbra is várom mindenki véleményét, ötleteit a ***Szy.Gyorgy@linuxvilag.hu*** címre!

---

absztrakció – elvonatkoztatás.

abstraction layer – elvonatkoztatási réteg; van, aki rövidít és az elvont szót használja.

address translation (NAT) – (hálózati) címfordítás; az egyik általános felhasználási területe az álcázás.

administrator – rendszergazda, felügyelő, üzemeltető stb.; lásd még network administrator.

array → lásd a telepnél.

bridge – híd (két – akár különböző típusú – hálózatot összekötő gép).

cache – gyorstár; sok értelemben használják; lényege, hogy olyan ideiglenes tároló, ahonnan gyorsabban érhető el valamilyen távoli erőforrás, amennyiben többszöri elérésre kerül sor.

cluster → lásd a telepnél.

critical – kényes, fontos, elsődleges, alapvető; sajnos gyakran a kritikus szót használják, többnyire hibásan. Néha különösen nehéz lefordítani, például a mission critical device többnyire „a feladat kulcseszköze”, vagy a „kitüntetett fontosságú eszköz” fordítást adják.

directive – irányelv, meghatározás; fordítóprogramoknál használatos például, bizonyos műveletek elvégzésének miként jét szabályozó beállítás.

directory service – címtárszolgáltatás; ilyen például az LDAP, de van címtármegoldása a Microsoftnak vagy a Novellnek is.

farm → lásd a telepnél.

gateway – (hálózati) átjáró.

handheld machine – lásd PDA.

interface – illesztő, illesztőfelület, felület, csatoló; nagyon sok értelemben használják.

kernel – rendszermag vagy röviden mag.

masquerading – álcázás.

mount – befűz; távoli fájlrendszereket fűzünk be a helyi rendszerfába, ezután a felhasználók ugyanúgy használhatják (megkötések mellett), mintha helyi fájlrendszer volna. Ellenművelete a kifűz vagy lecsatol.

network administrator – hálógazda; van, hogy hálózati rendszergazdának vagy viccesen hálófelügyelőnek hívják.

PDA – tenyérgép; egy fiatal ága a gépek családfájának, nap mint nap találnak ki új kütyüket, és mivel a piacról élnek, a gyártó mindegyik termékére kitalál valami új nevet. Lényegében a zsebtitkárok, a kézigépek, a marokgépek, a zsebnoteszek és az egyéb nevek mind ezt a kategóriát jelölik.

plugin – bővítmény.

point of sale device – eladási pont?

process – folyamat.

regex – lásd: regular expression.

regular expression – szabványos kifejezés; van valakinek találó rövidítése, olyasmi, mint az angolban a regex?

repeater – (hálózati) ismétlő, jelismétlő.

route – útvonal.

router – útválasztó.

routing table – útvonaltábla.

session – munkamenet.

shell script – héjprogram, parancsfájl.

site – telephely, webhely; több értelemben használt szó, igazából rövidítés. Magyarul a „hely” rövidítés értelemzavaró, ezért gyakran a hosszú változatot használjuk.

socket – foglalat.

stuff – töm; e havi „csodabogarunk”, jelentése: valamilyen eredetanyag (például egy képlemez) lementése a gépre egy kívánt formátumban. Mind az angol, mind a magyar szó szlengkifejezés (forrás: BME kollégiuma).

telep – több gépből felépített rendszer, mely egységes adatfeldolgozóként képes működni. Gyakran használnak telepeket nagy számításigényű feladat – például grafikai leképezések, csillagászati vizsgálatok, matematikai kutatások – elvégzésére. Telepeket felépíthetnek fürtözéses módszerrel is (array, matrix stb.), amikor az egyes elemek nem rendelkeznek önálló erőforrásokkal (saját memória, háttértár stb.), vagy önálló gépek különböző típusú összekapcsolásával (cluster, farm stb.).

thread – szál.

toolkit – eszköztár.

tunnel – alagút; hálózati megoldás, mikor egy protokollal működő kapcsolatban kiépítenek egy beágyazott átviteli lehetőséget, mely akár más típusú protokollt használ. Például A-ból kell E-be juttatni IPX-forgalmat, A-B-C-D vonalon, de C és D közötti kapcsolat csak IP-protokollal használható. Ekkor C és D között felépíthetünk egy IP alatti IPX-alagutat, mely képes ellátni az igényelt feladatot.

unload – az egyik „kemény diónk”. A load szót mindenki betöltre fordítja, ennek ellenművelete lehet az ürítés vagy az eltávolítás (például egy modul ürítése a memóriából).

version – változat.

Beköszöntő

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

Munka a hálózaton

Munka a hálózaton összefoglaló címmel jelenik meg negyedszázadik számunk, aminek talán az a legfontosabb apropója, hogy bizony ma már egyre több helyen komoly munkavégzésre használják ezt a nyílt operációs rendszert is. Igaz, egy irodában, raktárban, vagy más munkahelyen teljesen más igények merülnek fel, teljesen mást várnak el a dolgozók, mint mondjuk egy webkiszolgáló esetén, és a nyílt rendszereknek még komoly fejlődésen kell átmenniük, de ma már nyugodtan ajánlhatjuk irodai munkaeszköznek a GNU/Linuxot. Ugyanakkor van egy háttéroldal is, a kiszolgálók, amelyeknek csendben és megbízhatóan kell dolgoznunk, hiszen – bár az irodában nem látni őket, mégis – egy hibás vagy lassú kiszolgáló a vállalatot alapjaiban rengetheti meg. Emellett egyre-másra merül fel az igény „biztonságos" kiszolgálókra, egymást kiváltó, hibatűrő rendszerekre is, amelyek régen csak az óriásvállalatok kiváltságai voltak. Szerencsére kedvencünkkel akár hétköznapi gépekből is kiépíthetünk ilyen rendszereket – az ilyen átalakításokhoz találunk adalékokat e havi számunkban. Mivel a múlt hónapban feldobott LDAP-témakör fontosnak tűnt, még egy kört futunk a címtárak háza táján, sőt ***Lippai Gergő*** is küldött egy „átállás utáni helyzetjelentést".
Emellett gyakran előfordul a csoportmunka támogatásának kérdése, mely gyakran a következőre egyszerűsödik le: ki tudja-e váltani az Exchange-kiszolgálókat valamilyen GNU-rendszer? Az igény létező, sőt sokan akár fizetnének is egy rendszerért, csak ne kelljen külön MS operációs rendszert és kiszolgálóprogramokat venni, üzemeltetni. Szerencsére több megoldás is létezik, az egyik egy klón használata (ezek általában fizetős termékek, a „kereskedelmi Linuxok" közül több is forgalmaz ilyen kiszolgálót), a másik az igény átfogalmazása. A téma után érdeklődőknek ajánlom a 25. oldalon található „A Microsoft Exchange működése és helyettesítése" című cikket.
Azért mindenkit óva intek! Éppen a hétvégén tapasztaltam meg a saját bőrömön, hogy az ember gyakran lényegesen gyorsabban szeretne rendszert váltani. Történt ugyanis, hogy elhatároztuk, lecseréljük a már hosszú ideje köhögő-prüszkölő adatbázis-kiszolgálónkat (több hete fura hangokat hallat a merevlemez), de ha már azt átalakítjuk, akkor a tűzfalat is rendbe rakjuk, sőt a hálózatot is átépítjük stb. – szóval rendet teremtünk. A kiszolgálóról kiderült, hogy a táp és a merevlemez is hibás (a gép a régi táppal már-már nem is volt hajlandó újraindulni). A helyzet akkor kezdett komolyra fordulni, amikor vasárnap este 9-kor ültünk egy kupac vas tetején, és még semmi nem működött…
De evezzünk vissza vidámabb vizekre. Végre komolyabb cikket közölhetünk az UHU Linux háza tájáról: ***Koblinger Egmont*** meséli el nekünk a 12. oldalon, hányféle változást eszközöltek még az új UHU kiadása előtt.
Már csak egyetlen témával szeretnék előhozakodni, mielőtt átadom helyemet a remek cikkeknek, ez pedig az OpenOffice.org programcsomag magyarítása. Ma már szinte természetes, hogy GNU/Linux alatt az irodai munkákat az OpenOffice.org látja el. Magyar helyesírás-ellenőrzéssel ellátható, magyar a menüje stb. Igaz, sok hija van még a fordításnak. Pont ezek a hiányosságok ösztönöznek, hogy mindenkit arra buzdítsak, ha van kedve, lelkesedése, jöjjön el február 7-én a – már-már rendszeres – maratoni hétvégére (a bővebb írást lásd a 36. oldalon).
Mindenkinek kellemes időtöltést kívánok!

Programvadászat

Evolution

Mostani korongunk legterjedelmesebb darabja a Ximian cég Evolution programja. Ez a Outlook Express linuxos megfelelője, vagyis mindent egy helyen intézhetünk. Felülete is nagymértékben hasonlít az előbb említett Microsoft programhoz, így az átállás szinte zökkenőmentesen megtehető, a felhasználó kevéssé érzi majd idegennek a felületet. Nézzük, mit is kapunk kézhez ennek a programnak a telepítésével:

- Egy ügyes kis levelezőprogramot POP3-, IMAP-támogatással, azonban vigyázzunk, ha az IMAP előnyeit szeretnénk kihasználni, mivel elküldött leveleinket nem az IMAP kiszolgálóra menti, hanem a ***Helyi mappák->Elküldve*** mappába, ami bizony idegesítő lehet, ha más programból – mondjuk egy webmailből – is el szeretnénk érni a levelezésünket. Az ***Eszközök-> Beállítások-> Postafiók ->Szerkesztés*** menükön keresztül eljuthatunk az „Evolution Postafiók-szerkesztő”-be, ahol a ***Defaults*** menü alatt beállíthatjuk, hogy a hálózaton lévő ***INBOX.Sent*** mappába mentse ezeket a leveleinket.

- Határidőnaplót, ami riasztásra képes a megadott időpontban.
- Feladatlista-„észbentartót” szintén találhatunk, ez a feledékeny embereknek nagyon jó, de ha valaki egyszerre feledékeny és lusta, biztos nem fogja használni ezt a lehetőséget.
- Az én nagy kedvencem az ***Összefoglaló***, ezzel a képernyővel indul a program. Ide mindenféle okosság ki van téve – ezekben a hideg, hóeséses napokban bizony jó volt ránézni az időjárás-jelentésre, és arra gondolni, hogy „Bárcsak Szombathelyen laknék!”, amikor Budapesten -7 fok és erős hóesés volt, Szombathelyen pedig derült ég és +5 fok. Első nap nem is figyeltem fel igazán erre, de a következő nap elkezdtem gyanakodni, a harmadik napon pedig kénytelen voltam belátni, hogy Szombathelyen valószínűleg nyáron is ilyen idő lesz. Természetesen ezt a részt is személyre szabhatjuk, választhatunk számos hírcsatorna, a világ bármely táján fellelhető települések időjárása (Magyarországról még Pápa jöhet szóba az előbbi kettő mellett), és egyéb finomságok közül.
- Címtárat, ami leginkább egy névjegykártyatartóhoz hasonlít, és helyesen kezeli a magyar ékezetes betűket.

Ami számomra egy kicsit furcsa volt, az az, hogy nem mindenütt lett magyarítva, ezért helyenként (legfőképpen a beállításainál) bizony elkél az angol nyelvtudás.
A korongon a program Debian, Red Hat, SuSE és Mandrake Linuxhoz található meg, illetve a program forráskódja is felkerült.

Magazin

A Magazin könyvtár is hatalmasra hízott, ami leginkább a FreeS/WAN könyvtárnak köszönhető, ahová a 29. oldalon lévő cikkhez tartozó különféle Red Hat-rendszermagok kerültek be, előre fordítottan és forráscsomagban (természetesen ezek a rendszermagok bármelyik Linux-kiadáshoz használhatóak).
A HALDAP könyvtárban a „Magas rendelkezésre állású LDAP” (42. oldal) forráskódjai és beállítófájljai kaptak helyet.
A PostgreSQL adatbázis-kezelő használata 2. részének (58. oldal) forráskódjai egyszerű szöveg- és rtf formátumú fájlban is a CD-re kerültek, mivel a szövegfájlokban a magyarázatoknál az ékezetek könnyen elveszhetnek.
A Screen program forráskódját hasznosíthatják mindazok, akiknek Linux-kiadása nem tartalmazza ezt a nagyszerű csomagot – használatához a 38. oldalon találhatnak segítséget.
Mindazok, akik gyorsan és hatékonyan szeretnének kisméretű grafikus programokat fejleszteni, az FLTK csomag lehet a megfelelő felület. A fejlesztői környezet és a 46. oldalon kezdődő cikkhez tartozó forráskód, illetve a futtatható képnézegető program az ***fltk*** könyvtárban lelhető fel.

Videotelefon? Nem akadály! Marcel bevezet bennünket az internetes képtelefónia rejtelmeibe, amihez mi a GnomeMeeting programot adjuk – a szükséges kiegészítőkkel, különféle csomagolásban (70. oldal).

OpenOffice.org.deb

Bizony, furcsa ez a fentebbi cím, azonban a 11. oldalon magyarázatot kaphatunk rá és arra, hogy és mi végett lehetünk rá büszkék. Debianos telepítőkészlet OpenOffice.org-hoz, magyar csomagolásban. A legújabb üzembiztos rendszermag fejlesztési foltja is felkerül a CD-re.

Sajnálatos módon a CD-mellékletekkel többször is előfordult, hogy az 52×-esnek kikiáltott meghajtókban szétrobbantak, ezért kérjük olvasóinkat, hogy a saját és meghajtójuk biztonsága érdekében 32×-esnél nagyobb meghajtókban ne használják a korongokat.

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

Hitachi-újdonságok

A Hitachi nem kevesebb, mint 4 GB adatot szeretne – nemrég még IBM név alatt futó – Microdrive meghajtóinak legnagyobb tárhelyet kínáló példányaira zsúfolni. A mindössze 1"-os meghajtóknak – a világ legkisebb merevlemezeinek – a fejlesztése számos mérnöki kérdést felvet, ám a tervek szerint ezeket 2003 őszére sikerül megoldani. Az új apróságok egyrészt kisebb alkatrészeket kapnak, az író-olvasófej például feleakkora lesz, mint elődje, illetve az adatsűrűségüket is megnövelték. Ennek is köszönhető, hogy a Microdrive meghajtók messze gyorsabbak, mint az azonos területre készülő egyéb adattároló megoldások.
A Hitachi az üzleti felhasználók számára is tartogat újdonságokat, ilyen az Ultrastar 15K73 merevlemez, amely – mint neve utal rá – 73 GB kapacitású, üvegalapú korongjai pedig 15 000 fordulatot tesznek meg percenként. Az új meghajtók folyadékcsapágyas motort kaptak, a számítógéphez Ultra320 SCSI vagy 2 Gb FC-AL felületen keresztül csatlakozhatnak. Néhány teljesítményadat: 2 ms átlagos lappangási idő, 3,9 ms fejléptetési idő, akár 77,2 MB/sec fenntartható adatátviteli sebesség.
Újdonság, hogy a hordozható számítógépekben megszokott 2,5"-os merevlemezek helyett-mellett a Hitachi a továbbiakban 1,8"-os meghajtókat is kínál.
A Travelstar Compact Series C4K40 az új sorozat első tagja, az áttérést megkönnyítendő a 2,5"-os meghajtóknál megszokott csatlakozókkal látják el. Tartalmát tekintve semmivel sem marad el legkorszerűbb társaitól, hiszen üvegalapú korongokat tartalmaz, amelyek adatsűrűsége az új Microdrive meghajtókéival egyezik meg.
➲ http://www.hgst.com

A Motorola, az Avaya és a Proxim

E cégek közös tervezet keretében próbálják összeházasítani a mobiltelefonos és a vezeték nélküli hálózatokat. A cél az, hogy a mobiltelefon-tulajdonosok – Motorola telefonnal és Avaya programmal felszerelkezve, a Proxim által fejlesztett eszközök segítségével, átlátszó módon – a mobiltelefonos és a vezeték nélküli helyi hálózati kapcsolatokat egyaránt használhassák. Az így létrejövő háttérrendszer segítségével a felhasználó folyamatos hang- és adatkapcsolatot létesíthet, akár a vállalati, akár a mobiltelefon-hálózathoz csatlakozik, akár valamilyen nyilvános vezeték nélküli hálózattal lefedett területen (például repülőterek, internetkávézók, szállodák) tartózkodik. Hasonló területre merészkedik az észak-amerikai inCode Telecom-Bell Canada társulás. Az ötlet egyszerű: a forgalmasabb helyeken lévő telefonfülkéket egy vezeték nélküli hálózati hozzáférési ponttal látják, így a fülre kb. 100 méteres körzetében hálózati lefedettséget tudnak biztosítani.
Ha valaki például útközben le szeretné tölteni a leveleit, és megfelelő hordozható géppel rendelkezik, csak egy telefonfülkét kell keresnie, és máris internetkapcsolatot létesíthet.
A megoldás a szolgáltató számára is egyszerű és olcsó, hiszen a telefonfülke eleve rendelkezik valamilyen hálózati kapcsolattal, így viszonylag kis bővítéssel vezethet be teljesen új szolgáltatást.

Terjeszkedik a MontaVista

A beágyazott Linux-terjesztéséről ismert MontaVista bejelentette, hogy rendszerének Consumer Electronics Edition 3.0 változatával belép a fogyasztói elektronikus készülékek piacára.
A fejlesztők munkáját a célterület sajátos igényeihez igazított eszközökkel, illetve hamarosan grafikus kiegészítőkkel és Java-környezettel is segíti a MontaVista.
A háztartási készülékek eddig különálló, adott célfeladatra használható eszközök voltak, de a kor divatját követve egyre összetettebb, hálózati kapcsolattal rendelkező berendezésekké válnak. Ebbe a környezetbe remekül illeszkedik a MontaVista dinamikus energiakezelésre és fejlett hálózattámogatásra képes, megbízhatóságát tekintve is élvonalbeli operációs rendszere, amelyet a növekvő igény hatására teljes, az IBM WebSphere Studio termékére alapuló Java fejlesztői és futtatási környezettel is kiegészítettek. Mivel a linuxos alaphoz számos alkalmazás készen szerezhető be, a készülékek gyártói felgyorsíthatják az eszközök fejlesztését, és rövidebb idő alatt piacra dobhatják újdonságaikat.
Már most is több mint száz készülékben – zsebtitkárok, internetes rádiók, otthoni átjárók, karaoke-gépek – találni Linuxot, de hamarosan digitális tévékben is feltűnik a MontaVista beágyazott operációs rendszere.
➲ http://www.mvista.com

Digitális fényképkeret

Megvásárolható az első USB-csatlakozással ellátott digitális fényképkeret – már ha szabad ilyen módon lealacsonyítani a MemoryFrame nevű készüléket.
Az eszköz egy elegánsabb fényképkerethez hasonlítható, a közepén azonban nem papírkép, hanem egy 5,6"-os TFT kijelző található, amely a memóriájában – 16 MB RAM és 8 MB ROM – tárolt képeket 640×480-as felbontásban jeleníti meg. A készülék állítva és fektetve is elhelyezhető, illetve falra szerelhető, kezelőszervei a hátlapjára kerültek. A fényképeket USB 1.1 felületen keresztül tölthetjük rá, akár közvetlenül a fényképezőgépről is. Üzleti környezetben egy Kensington aljzat segít abban, hogy a digitális keret ne csak a biztonsági őr háta mögött lehessen biztonságban.
➲ http://www.pacificdigital.com

Karpov-Kaszparov játszma 3D-ben

Anatoly Karpov 2,5–1,5 arányú győzelmet aratott *Garry Kaszparov* felett az első Extreme 3D megoldással közvetített sakkmérkőzésen.

Karpov ugyan rosszul indított, az első játszmát elvesztette, de rövidesen ellenfele fölé kerekedett. A két mester játékát először közvetítették három dimenzióban, élőben és az Interneten keresztül. A műszaki hátteret az X3D System biztosította, a nézők a megfelelő alkalmazással és 3D LCD szemüveggel felszerelkezve virtuális térben követhették a játszmát.
➲ http://www.x3dworld.com

Beágyaz a Transmeta

A Transmeta energiatakarékos, x86 alapú beágyazott alkalmazásokhoz ajánlott beágyazott processzorral próbál nagyobb szeletet hasítani magának a processzorpiacról. A Crusoe SE lapkák egy LongRun nevű energiakezelő megoldással vigyázzák saját épségüket, amely folyamatosan a megfelelő értékre állítja a processzor órajelét és tápfeszültségét, illetve figyeli a hőmérsékletét. A beágyazott készülékekben a mostoha körülmények miatt sokszor bajos az aktív hűtés használata, ám a Transmeta akár 100 °C hőmérsékletet is elviselő lapkái hűtőventilátor nélkül is jól érzik magukat. A Crusoe SE 667, 800 és 933 MHz-es órajellel lesz kapható.
➲ http://www.transmeta.com

Bluetooth alapú hálózat otthonra

Az amerikai háztartásokban szinte alapvető tartozék a vezeték nélküli telefon. A Wave Industries ezt a szokást próbálja meglovagolni Bluetooth alapú, háztartásokba szánt vezeték nélküli telefonkészülékeivel. A rendszer érdekessége, hogy a szabványos rádiós összeköttetésnek köszönhetően nemcsak telefonálni lehet vele, de egyéb eszközök, így zsebtitkárok, hordozható számítógépek, mobiltelefonok is csatlakozhatnak rá. A cég kiegészítő termékekkel igyekszik vonzóbbá tenni megoldását, például Bluetooth fülhallgatója megfelelő mobiltelefonnal és az új otthoni rendszerrel egyaránt használható, USB-Bluetooth-csatolója pedig asztali számítógépekhez készült.

A Bluetooth kapcsolatok csak tíz méteres távolságon belül használhatók, a cég azonban egy felerősített változatot használ, ami szabad térben 260, házon belül pedig 50 méterig hatékony. A központi egység legfeljebb hét kézi beszélőt tud kiszolgálni – ezek egyébként ikonvezérelt menüvel, grafikus LCD-vel rendelkeznek, csengőhangjuk programozható, illetve saját telefonkönyvet tárolnak –, a számítógépek pedig a központ beépített 56 kb/s sebességű modemjével internetezhetnek.

➲ http://www.olympiaphones.com

James Bond és a PolyLED kalandjai

Ha másért nem, hát azért érdemes megnézni a legújabb James Bond-filmet, hogy láthassuk a Philips PolyLED (polimer alapú szerves fénykibocsátó dióda) kijelzőinek egy példányát. A főhős nem kevésbé fontos használati tárgyát, mint a borotváját kell jól megfigyelnünk, ha látni akarjuk a cég első kereskedelmi forgalomban kapható PolyLED kijelzőjét. A James Bond-filmekben a főszereplők kezdettől fogva az adott kor különleges műszaki újdonságait vetik be kalandjaik során, és mint láthatjuk, a szuperkém borotvából is csak a legújabbat veszi kezébe. A PolyLED kijelzők különösen vékonyak és könnyűek, fényerejük, kontrasztjuk és láthatósági szögük mégis viszonylag nagy, fogyasztásuk és válaszidejük ellenben alacsony, jól tűrik a nagy mértékű hőmérséklet-ingadozásokat, így elsősorban hordozható vagy kisméretű eszközökben találkozhatunk majd velük.

Tüzelőanyag-cella háztartásokba

A Mitsubishi bejelentette, hogy – bár egyelőre csak kipróbálási célokra – megkezdte 1 kW teljesítményű, háztartásokba fejlesztett tüzelőanyag-cellájának szállítását. A készülék normál vezetékes gázzal működik, amelyből a Tokyo Gas által fejlesztett eljárással vonja ki a hidrogént. Belső védelmének köszönhetően tulajdonosa szükség szerint kapcsolhatja be vagy ki. A cég tervei szerint a cellát legalább két évig próbálják ki, elsősorban megbízhatósági és biztonsági szempontból, az üzletekben legkorábban 2005-ben lehet majd megvásárolni.

BioAPI Linuxra is

A BioAPI Consortium bejelentette a BioAPI Linux/Unix-rendszerek alá készített átültetésének próbaváltozatát.

Egyelőre a társulás tagjai végzik a kipróbálást, de hamarosan a nyilvánosság számára is elérhető lesz az első referenciamegvalósítás. Az API Windows alá már létezik, de támogatói – köztük az amerikai Kereskedelmi Minisztérium ügynöksége, a NIST is – szeretnék minél szélesebb körben elérhetővé tenni, és minél több Unix-változatot támogatni. A jelenleg támogatott Linux és Solaris mellé várhatóan a BSD, a HP-UX és az AIX is felsorakozik.

A BioAPI nyílt alkalmazásprogramozási felület, amelynek segítségével az alkalmazások biometriai eszközök széles körével – ujjlenyomat-azonosítók, hang- és arcfelismerők, íriszletapogatók stb. – létesíthetnek kapcsolatot. A felület 1.1-es változata, illetve a windowsos referenciamegvalósítás 2001 márciusában jelent meg, és szabadon, forráskódként is letölthető a társulás honlapjáról; ez egyébként a linuxos változattal is így lesz. A BioAPI fejlesztését az iparág számos szereplője támogatja, köztük biometriai eszközöket gyártó cégeket, rendszerintegrátorokat, számítógépgyártókat és végfelhasználókat egyaránt találunk. A felületet 2002 februárjában az ANSI hivatalos szabványaként fogadta el, és hamarosan várhatóan ISO/IEC szabványként is jóváhagyják. Nemcsak számítógépeken, de beágyazott eszközökön – például mobiltelefonokban – is használható, illetve nagyméretű rendszereket is képes kiszolgálni.

➲ http://www.bioapi.org

SGI Altix 3000

Az SGI bejelentette SGI Altix 3000 nevű kiszolgáló- és telepcsaládját. Az új gépek Intel Itanium 2 processzorokat tartalmaznak, Linux operációs rendszert futtatnak, és az SGI szuper-számítógépes megoldásaival próbálnak vonzókká válni a leendő vásárlók számára. Az utóbbiak leginkább tudományos kutatásokat, mérnöki munkákat vagy egyéb, különösen nagy számítási teljesítményt igénylő feladatokat végző szakemberek és intézmények közül kerülnek majd ki.

Az SGI Altix 3000 egy példányban futtatja az operációs rendszert, amely legfeljebb 64 darab processzorral és 512 GB memóriával gazdálkodhat. A gépek beépített teleptámogatással rendelkeznek, az SGI telepmegoldása a hagyományos összeköttetéseknél akár kétszázszor gyorsabb lehet, így semmi akadálya annak, hogy több száz vagy több ezer processzor dolgozzon együtt. Az SGI megoldásának köszönhetően a létrejövő telep csomópontjainak memóriája a teljes telepre kiterjedően, operációs rendszerek között is megosztható; az így létrejövő szuperszámítógépen az összetett modellek – mint például a Föld időjárási modellje – is viszonylag könnyen, apróbb részekre való bontás nélkül kezelhetők.

➲ http://www.sgi.com

3D-képernyő a Matsushitától

A Matsushita Electric Works hamarosan félgömb alakú, 1,8 méter átmérőjű 3D-kijelzőt dob piacra. A készülék láthatósági szöge vízszintesen 140, függőlegesen 90 fokos, fényereje 2000 ANSI lumen, méreténél fogva pedig akár életnagyságú – saját programjával torzításmentessé tett – 3D-képek megjelenítésére is alkalmas. Az eszközt tervezési feladatoknál, például járművek belső terének kialakításakor lehet hasznosítani, de különféle modellezési alkalmazásokban és a szórakoztatóiparban is megtalálja helyét.

Intel Centrino

Az Intel új márkaneve a Centrino, amely a cég történetében először nem valamilyen terméket, hanem termékegyüttest – nevezetesen a hordozható számítógépekbe készülő megoldásokat – takar. Az Intel Centrino magába foglalja a cég új mobil (Banias kódnevű) processzorait, a hozzájuk készülő lapkakészleteket, a vezeték nélküli hálózatok elérésére szolgáló csatolókat, illetve magát az energiatakarékos, könnyű és kisméretű készülékek építésére létrehozott környezetet.
A Centrinónak saját logója készült, amely tartalmazza a jól ismert Intel Inside feliratot, ugyanakkor szárnyakat is kapott ami a repülésre, az előretekintésre, a mobilitásra utal.

Elérhető az OVAL

A Mitre Corporation elkészült az Open Vulnerability Assessment Language (OVAL) leíró nyelvvel. Az OVAL a várakozások szerint széles körben elfogadott, a számítógépek biztonsági hiányosságainak leírására használható szabvánnyá válik.
Biztonsági hiányosságnak ebben az esetben azt a pontot kell tekinteni, amelyen keresztül egy betörő behatolhat a számítógépre. Ezeknek a pontoknak a felderítésére többféle program is létezik, ám messze nem egységes a működésük, és sokszor még abban is eltérőek, hogy mit tekintenek biztonsági hiányosságnak – ebből pedig egyenesen következik az, hogy eredeti céljukkal ellentétben nem tekinthetők megbízható megoldásnak a hiányosságok számba vételére.
Az OVAL egységes, általános szempontrendszert állít fel, amelynek alapján a szakértők megbízható módszerrel vizsgálhatnak át egy-egy rendszert. A megválaszolandó kérdéseket SQL nyelven írják le, azután a vizsgálati környezet akár biztonsági programokba is beépíthető.
Az OVAL a Common Vulnerabilities and Exposures (CVE) „szótárra” épül, amely szabványosított névvel és leírással veszi számba az ismert biztonsági réseket, hiányosságokat. Jelenleg a Windows NT 4.0, Windows 2000, Solaris 7 és 8 operációs rendszereket támogatja, a Red Hat Linux támogatása egyelőre részleges.
➲ http://oval.mitre.org

Memory Stick PRO

A SanDisk bemutatta első Memory Stick PRO kártyáit. Az újfajta memóriakártyák több újdonságot vonultatnak fel régebbi társaikhoz képest, bár ezeket a lehetőségeket a kártyát megtöltő készüléknek is támogatnia kell.

A PRO kártyák akár valós idejű, DVD-minőségű mozgóképlejátszásra, vagy -felvételre is használhatók, illetve bármely olyan alkalmazásban jó szolgálatot tehetnek, ahol nagy mennyiségű adatot kell gyorsan mozgatni.
A Sony–SanDisk-együttműködésben fejlesztett kártyák további lehetőségekkel is támogatják a személyes adatok védelmét – kereskedelmi felhasználók számára a MagicGate megoldással másolásvédelmet is nyújtanak –, illetve felvétel közben történő véletlen eltávolításuk vagy tápkimaradásuk sem okozza a rajtuk található fájlrendszer sérülését.
A Memory Stick PRO kártyák első körben 256, 512 és 1024 MB-os méretben jelennek meg, régebbi – jellemzően kék – társaiktól aranysárga színük különbözteti meg őket; méretüket és formájukat tekintve a két nemzedék tagjai megegyeznek.
➲ http://www.sandisk.com

Életet menthet az IBM pulzusmérője

Az IBM olyan kisméretű, viselőjének a szívverését figyelő eszközt mutatott be, ami vészhelyzet esetén mobiltelefonon keresztül is segítséget tud hívni tulajdonosához. A készülék Bluetooth kapcsolaton keresztül küldi el a vészjelet, így nemcsak mobiltelefonnal, hanem bármely Bluetooth-képességekkel felruházott eszközzel képes kapcsolatot teremteni. Hasonló, szívműködést figyelő készülékek már régóta léteznek, és általában a vizsgált személy csuklójára csatolják őket, ám ezek a pulzus mérésén kívül másra nem nagyon alkalmasak. Az IBM kisméretű, könnyű készülékét további célokra is szeretnék majd felhasználni, így például edzőcipőbe építve követhető és utólag elemezhető a sportolók mozgása.

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.

Cégvilág

Átfogó munkaállomások a HP-tól

A HP új, kedvező árú munkaállomások bevezetésével tette még hangsúlyosabbá portfoliója egységesítését. A versenyképes árfekvésű, Intel Pentium 4 és Intel Xeon processzorral dolgozó HP-munkaállomások teljesítménye és programtámogatási háttere szinte minden felhasználói szükségletet kielégít, többek között a pénzügy, a számítógépes tervezés (MCAD/CAE), a digitális tartalomfejlesztés (DCC) és a térinformatika (GIS) területén.

A HP-től és a cég kiterjedt értékesítési partnerhálózatának tagjaitól beszerezhető új xw4000, xw5000, xw6000 és xw8000 személyi munkaállomások a Microsoft Windows 2000 Professional, a Windows XP Professional és a Linux operációs rendszert támogatják. Az új c3750 és j6750 HP műszaki munkaállomások HP-UX operációs rendszerrel és nagy teljesítményű PA-8700+ processzorral működnek. E rendszerek kialakításakor a hagyományos Unix-felhasználók által elvárt megbízható számítási és megjelenítési teljesítmény biztosítása volt az elsődleges szempont.

Sharp és IBM

A Sharp Electronic és az IBM összefogtak annak érdekében, hogy még április előtt piacra kerülhessen az első olyan Sharp tenyérszámítógép, amelyik Linuxon fut. Ez a szövetség tekinthető e téren az első próbálkozásnak a nagyobb gyártók részéről. A Linux alkalmazásának egyébként ugyanaz az oka, mint a nagygépes rendszerek esetén: ingyenes hozzáférhetősége segít alacsonyan tartani a készülék árát. Amerikában a Linuxot futtató PDA-k jelenlegi piaci részesedése egy százaléknál kevesebb, s ez az előrejelzések szerint 2006-ra is csak 3,5 százalékra fog növekedni. A Sharp Zaurus nevű termékével igyekszik kivenni a részét ebből a bővülésből. Nem is elsősorban a Linux révén kívánnak robbantani a piacon, hanem a Linux-Java párosítással (azaz javás programok Linuxon való futtatásával) szeretnék megnyerni maguknak a vállalati felhasználókat.

Kárpótlási jegy a Microsofttól

A Microsoft 1,1 milliárd dollár értékű peren kívüli egyezséget kötött azokkal a kaliforniai magánszemélyekkel és vállalatokkal, akik piaci uralommal való visszaélés, azaz indokolatlanul drágán árusított programjai miatt perelték be. „A per költséges elhúzódásának elkerülése érdekében kötött megállapodás jelentős lépés a függőben lévő száz másik egyesített ügy megoldását illetően is, és ahol csak lehet, megegyezésre törekszünk” – nyilatkozta *Brad Smith*, a Microsoft vezető jogtanácsosa. A Microsoft tavaly 660 millió dollárt különített el perekre.
A még a San Franciscó-i bíróság formális jóváhagyására váró egyezség értelmében a Microsoft a pertársaságot alkotó immár 13 millió egykori felperesnek számítógépek, programok és bizonyos perifériák vásárlására szolgáló olyan utalványokat ad, amelyekkel versenytárs cégek termékei is vásárolhatók.
Forrás: jogiforum.hu

Szuperszámítási szolgáltatás az IBM-től „igény szerint”

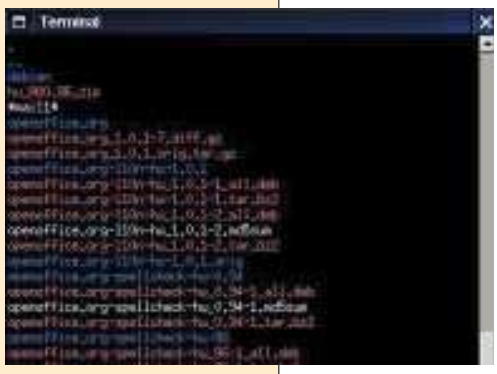
On-Demand stratégiája keretében az IBM bejelentette új, igény szerinti szuperszámítási kezdeményezését. Az új szolgáltatás révén a felhasználók „mentesülhetnek” a nagykiszolgálók vásárlása és karbantartása alól, amennyiben meghatározott munkák elvégzésére rövid távú szerződést kötnek az IBM-mel. A cég ügyfelei vagy a Power4 processzorokra épülő Unix kiszolgálóhálózatot vehetik igénybe, vagy az Intel Xeon processzoraira épülő linuxos szekrénykiszolgálót (rack server). *Dave Jursik*, a Linux-telepek ügyében illetékes alelnök elmondta, hogy a ciklikus munkaszakaszok szerint dolgozó cégek, például a filmstúdiók gyakran vásárolnak kiszolgálókat, sőt olykor szupergépeket is. Két feladat között ezek a gépek kihasználatlanul maradnak, de karbantartásuk így is költséges. Az e-Business on Demand szolgáltatás lehetővé teszi, hogy az ügyfélcég az adott programjához szükséges szuperszámítási erőforrást használja föl, miközben megszabadul a háttér kiépítésétől és karbantartásától – megállapodnak a projekt időtartamáról és az IBM által biztosítandó erőforrások mennyiségéről.
➲ http://www.ibm.com

Az Apple kiadta az X11 Mac OS X-re készült változatának bétáját

Az Apple a San Franciscó-i Macworld konferencián és kiállításon újabb lépést tett a hagyományos Unix-rendszerek felé, a cég ugyanis bejelentette az XFree86 Mac OS-re készült változatának első próbaváltozatát. Nemrégiben már bejelentettek egy nyílt forráskódú fejlesztésekre épülő új programot, a Safarit: a webböngésző az egyéb Unix-szerű rendszerekről már jól ismert KDE grafikus környezet egyik fontos elemét, a KHTML-t használja fel a weboldalak értelmezéséhez és megjelenítéséhez. A most bemutatott Mac OS-es X11-változat az XFree86-ra épül. A már több mint tízéves grafikus alaprendszer (X kiszolgáló, programkönyvtárak, alapvető alkalmazások) a nyolcvanas évek közepén megalkotott X Protocol legismertebb megvalósítása. Napjainkban a legtöbb Unix-szal együttműködő rendszer – így az összes neves Linux-terjesztés grafikus felülete is – az XFree86 különböző kiadásait használja fel. X11 for Mac OS X a Macintosh-felhasználók számára lehetővé teszi, hogy az eddig csak Linuxon vagy Unixon elérhető grafikus programokat ezentúl az Apple operációs rendszerén is futtassák. A programcsomag kihasználja a Mac OS lehetőségeit, és a natív Aqua-alkalmazások mellett olyan legendás linuxos programok is felbukkanhatnak, mint például a Gimp.

Kelényi Attila (attila@kiskapu.hu)

OpenOffice.org Debian-csomagban



A csomag elkészítésére az ösztönzött, hogy számos magyar Debian-felhasználóval együtt megkértem azokat, akik *.deb* csomagokat tudnak készíteni, hogy valaki hozza létre ezt a csomagot. Mivel erre nem került sor, egy karácsony utáni napon, egy családi összejövetelt követően, késő este ittam egy jó erős kávét, és mivel nem tudtam elaludni, valami értelmesnek próbáltam meg eltölteni az időt. Így született meg a saját deb-csomagom. Hozzáfogtam, hogy megtanuljam, hogyan is kell deb-csomagokat készíteni (ha valaki szintén meg szeretne próbálkozni vele, a ➲ http://people.debian.org/~jaldhar/make_package1.html címről célszerű elindulnia). Ennek az lett az eredménye, hogy hajnali öt órára be is fejeztem a magyarítás első változatát.

A csomag létrehozása során nehézséget jelentett, hogy nem tudtam pontosan olyan módon összerakni, mint ahogyan azt a hivatalos csomagokban megszokhattuk. Egyrészt már a készítői sem vállalták el, másrészt rengeteg időbe került volna. Ennek ellenére másnapra sikerült egy mind a formáját, mind a tartamát tekintve a hivatalos csomagoknak teljesen megfelelő csomagot összerakni. A helyesírás-ellenőrző viszont pofonegyszerű volt, csak fogtam a Debianban lévő német magyarítás, és egyszerűen átírtam.

Ezek után jött a dolog terjesztés része. Nagyon örültem, hogy kész a csomag, viszont úgy gondoltam, nem leszek önző, és megosztom a többi, e csomagra várakozóval is (elvégre erről szól a nyílt forráskód). Feliratkoztam a magyar Debian-levelezőlistára, valamint a magyar OpenOffice.org-levelezőlistára, ezenkívül hírként is elküldtem az általam ismert, hírbeküldési szolgáltatással is rendelkező, Debiannal is foglalkozó oldalakra (így értesülhetett róla a Linuxvilág is). A ➲ http://hup.hu-n erős bírálatot kaptam, hogy miért pont egy erősen „warez” körökben ismert ingyenes tárhelyre töltöttem fel, amiért akkor még nagyon haragudtam, így visszanézve azonban ez csak előnyére vált a csomagoknak. Így mára állandó és biztonságos tárhelyük van, valamint `apt`-vel is elérhetők.

Elsőként „szogi” segített ki, aki a rendszermagforgatásról készített leírásom alkotásába is lelkesen besegített. (Ezt a dokumentációt még mindig fejlesztgetem, amennyire az időm engedi. Ha valakit érdekel, a ➲ http://www.tar.hu/balihb/kernelforgatas címen elérhető. A ➲ http://savannah.gnu.org-on sikerült regisztrálni egy projektet, így már én magam tudtam a csomagokat feltölteni. Ennek megfelelően a csomagok telepítése sokkal könnyebb lett. A */etc/apt/sources.list*-hez hozzá kell adni a következő sort:

```
# deb http://savannah.nongnu.org/
➥download/ooodeb-hu ./
```

aztán az

```
# apt-get clean
# apt-get update
# apt-get install openoffice.org-
➥spellcheck-hu
# apt-get -d install openoffice.org-
➥l10n-hu
# dpkg --force-overwrite -i
➥/var/cache/apt/archives/openoffice.
➥org-l10n-hu_1.0.1-2_all.deb
```

parancsokkal települ minden, oda, ahova kell. Ez csak a Sid esetén igaz, mert a Woodyban még nincs benne az OpenOffice.org, így Woodynál a telepítés idejére vagy át kell írni a *sources.list*-et Sidre, vagy hozzá kell adni egy woodys OpenOffice.org-tükröt a *sources.list*-hez (➲ http://www.linux-debian.de/openoffice/mirrors.html). A csomag csak a debianos OpenOffice.org-ra települ föl, ha tehát az OpenOffice.org az office.fsf.hu-n megtalálható telepítővel van telepítve, azt érdemes előtte letörölni. Az, hogy a magyarítást a `--force-overwrite` kapcsolóval kézzel kell telepíteni, ne tévesszen meg senkit – a hivatalos csomagokat is így kell telepíteni. Ha időm engedi (valamint sikerül megtanulnom a GPG használatát), ezek a csomagok valószínűleg a hivatalos Debianba is be fognak kerülni, úgyhogy lehet, hogy már nem kell sokáig szenvedni a *sources.list* módosítgatásával. Szerencsére úgy tűnik, hogy egyelőre folytatni tudom a csomag fejlesztését, így ha rajtam múlik, a Debian nem marad többet OpenOffice.org-magyarítás nélkül.

Hámorszki Balázs (balihb@cracker.hu)

UHU-Linux: egyedi csomagkezelő, friss csomagok

Némi késéssel ugyan, de e cikk megjelenésekor végre mindenki számára elérhetővé válik az UHU-Linux legfrissebb változata.

Az eredetileg tervezett megjelenéshez képest megközelítőleg egy hónapot kellett várnunk, mivel a csapat a felhasználók érdekében úgy döntött, késlelteti a megjelenést, hogy egy kiforrottabb rendszert nyújthassanak át a felhasználóknak. Megkerestem ***Koblinger Egmont***-ot, az UHU-Linux fejlesztői csapatának egyik oszlopos tagját, és két érdekes témáról kérdeztem. Egyrészt szerettem volna megtudni, ténylegesen milyen a sokat firtatott új csomagkezelő, az `uhu-pkg`, másrészt kíváncsi voltam az új terjesztés csomagjaira.

Gibizer Tibor: *Az rc1 megjelenését a hibajavítások időszaka követte, de közben már dolgoztatok az új „uhubuild"-en. Hol tart a fejlesztés most?*

Koblinger Egmont: Felismertük, hogy tartósan jó minőséget csak úgy lehet létrehozni, ha nemcsak magára a termékre összpontosítunk, hanem azokra a belső eszközökre is, amelyek segítségével a terméket elkészítjük. Az rc1-es kiadásig használt eszközökkel a hibák kijavítása igencsak nehézkes volt. Ezért ősszel az rc2 készítését nem közvetlenül a hibajavításokkal kezdtük el, hanem egy saját csomagkészítő rendszer, az uhubuild megtervezésével és megvalósításával. Kettős cél vezérelt minket: olyan hátteret kívántunk kialakítani, amely a saját munkánkat megkönnyíti és hatékonyabbá teszi, valamint egységesebb és magasabb minőséget ígér az elkészült csomagok és ezáltal a teljes terjesztés terén is. A tervezést és az első változat megírását követően a csomagokat elkezdtük átültetni az új rendszerre.
A folyamat során szerzett tapasztalatok figyelembevételével továbbfejlesztettük az uhubuild rendszert. Ősz végére jutottunk el oda, hogy mindkét célunkat maradéktalanul elértük, és ténylegesen elkezdhettünk foglalkozni a terjesztéssel – immár a réginél jóval nagyobb hatékonysággal. Természetesen az uhubuild nem befejezett projekt, és valószínűleg soha sem lesz az, számtalan ötletünk van még a továbbfejlesztésére.
Az uhubuild működéséről címszavakban: az `apt-get` segítségével egy `chroot` rendszert épít fel, amelyben pontosan azok a csomagok vannak telepítve, amelyeket az elkészíteni kívánt csomag fordítása során látni szeretnénk. Természetesen ezeknek a csomagoknak a legfrissebb változata található meg itt. Befűzi a */proc* és */dev* fájlrendszereket, valamint a lefordítandó csomag forrását, végrehajtja a `chroot` rendszerhívást, majd innentől kezdve egyszerű felhasználóként (nem rendszergazdaként) végzi a *.uhu* kiterjesztésű deb-csomag elkészítését – közbeiktatva számtalan kötelező lépést, amelyek a csomagok egységességét teszik lehetővé ilyen az, egységes jogosultság beállítása a súgóoldalara nézvést, FHS-ellenőrzés, egy-két jellemző hibára való rákeresés és még számos hasznos szolgáltatás). Akit érdekelnek a részletek, a leírást is megtalálja a csomagban.
Sokan úgy gondolják, hogy az UHU egy Debian alapú terjesztés, de ezt mindig ki szoktuk javítani arra, hogy valójában `dpkg` alapú. A kezdetek során tényleg sok mindent átvettünk a Debiantól, ma már azonban nagyon kevés maradt ezekből, és ha valamit más terjesztésekből kell átemelnünk, akkor is a Debian csak egy a jó néhány közül, amelyet megvizsgálunk. Csomagjaink továbbra is deb-csomagok, a `dpkg` csomagkezelőt használjuk, azonban az uhubuild rendszer szinte semmilyen hasonlóságot nem mutat a Debian csomagkészítési módszerével. Nem használunk „rules" fájlt, a „control" fájlt is parancsfájlok készítik el, tehát nem kézzel farigcsáljuk őket. Mivel a végeredmény most is deb-csomag, aki csak használni kívánja a rendszert és nem érdeklik a kulisszatitkok, nem fog gyökeres változást látni. A szemfülesebbek észrevehetnek egy-két kiegészítést az eredeti `dpkg`-hoz képest, ilyen például a „Homepage" mező a control fájlban, a csomagfordítási adatok, vagy a stat fájl a telepített fájlok tulajdonságairól; de a lényeg változatlan. Ide kívánkozik az `uhu-pkg` rendszer megemlítése is.

G. T.: *Az `uhu-pkg` a csomagkezelő szerves része vagy ez is önálló alkotás?*

K. E.: Valamelyest a csomagkezelő rendszerhez kapcsolódik az `uhu-pkg` bevezetése, bár lényegében önálló program. Körülnéztünk a terjesztések között, és a csomagok telepítése után elvégzendő teendőkre (felhasználók létrehozása, `ldconfig` futtatása, *info.dir* elkészítése stb.) vonatkozóan alapvetően kétféle megközelítést találtunk. Az egyik szerint minden csomag kézzel-lábbal elvégzi, amit szeretne, ezáltal néhány csomagnak hatalmas, átláthatatlan *postinst* parancsfájlja van, amiben nehéz hibát javítani, illetve sok csomagban megtalálható ugyanaz a kód, ami szintén a karbantartást teszi nehézkessé. A másik megközelítés az általában nagy, lassú és áttekinthetetlen globális beállító parancsfájl. Ilyenkor azonban óriási hátrány, ha ezt valaki elfelejti kézzel lefuttatni: a csomag esetleg nem lesz használható. Ezért egy olyan megoldást kellett kitalálnunk, ahol egyrészt a parancssori `dpkg` (`apt`, `dselect` stb.) használata is

bőven elegendő, nem kell további parancsot lefuttatni; másrészt viszont a csomag legfeljebb annyit tud magáról, hogy mit kell vele csinálni (de talán még ennyit sem), semmiképpen sem azt, hogy hogyan. Ha például az `opensshd` hoz magával egy `sshd` felhasználót, akkor az `opensshd` csomag csak annyit tudjon, hogy ezt a felhasználót létre kell hozni. A létrehozás mikéntjét viszont másvalaki tudja. Ez a másvalaki az `uhu-pkg` nevű parancsfájl, amely minden csomag `postinst`, `prerm` és `postrm` parancsfájljaiból meghívódik, majd elindítja a megfelelő parancsot.

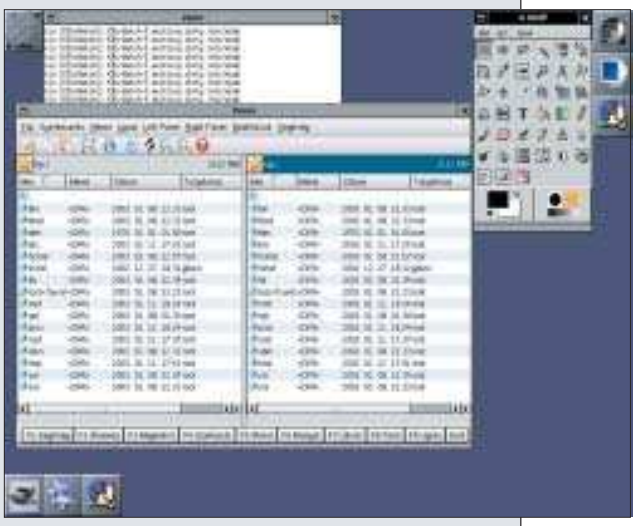
A `/sbin/uhu-pkg` a fő vezérlő parancsállomány. Annak a nagyon kevés feladatnak az elvégzésére szorítkozik, ami szigorúan kötelező lehet a csomag működése érdekében (felhasználó létrehozása, könyvtárak jogosultságainak a beállítása – ezzel a `dpkg` egyik hiányosságát pótoljuk – és az esetleges egyéni parancsállományok futtatása). Következő lépésben a `/sbin/uhu-pkg` a */usr/lib/uhu-pkg* alatti parancsfájlokat indítja. Az itt található parancsállományok végzik el azokat a feladatokat, amelyek más csomagok szempontjából lehetnek fontosak. Például a texinfo csomag ide helyezi azt a parancsfájlt, amelyik az *info.dir* fájlt fogja elkészíteni, a glibc ide teszi az `ldconfig`-ot futtató parancsállományt, továbbá az xfree86 ide rakja az ablakkezelők listáját, valamint a *fonts.dir* fájlokat készítő parancsfájlt és így tovább.

G. T.: *A most elmondottakból látszik, hogy komoly változások történtek a háttérben, de mit érzékel ebből a felhasználó?*

K. E.: Az „uhubuild" rendszernek köszönhetően jóval könnyebben tudunk haladni a programok változataival, így elmondható, hogy élen járó terjesztés lettünk. Szinte minden programból a legújabb elérhető üzembiztos változatot használjuk, például gcc-3.2.1, glibc-2.3.1, perl-5.8.0, kernel-2.4.20, kde-3.1-rc6, openssl-0.9.7 – a sort a végtelenségig lehetne folytatni.

G. T.: *Ezek valóban komoly változások. A csomagkezelő fejlesztése közben maradt időtök a számos felhasználói kérés teljesítésére is? Itt arra a rengeteg csomagra gondolok, amit a honlapotokon (➲ http://www.uhulinux.hu), illetve a levelezőlistátokon (uhu-kezdo@uhulinux.hu) keresztül kértek a felhasználók.*

K. E.: Természetesen minden igényt igyekeztünk kielégíteni, ami a rendelkezésünkre álló helyre befért. Ami a levelezőlistán szóba került, vagy amit a honlapon keresztül kértek, azoknak a legnagyobb része belekerült a terjesztésbe. Ezek között említhetem meg az alacsony rendszerigényű ablakkezelőket (IceWM, Window Maker és Blackbox), a NUM LOCK-ot bekapcsoló `numlockx` programot és a KDE kétpaneles `pcmm` fájlkezelőjét (amelyet le is fordítottunk magyarra), vagy éppenséggel az arenaportal.hix.com fórumhoz készült ügyfélprogramot. Pótoltuk a hiányzó alapcsomagokat, ezek például a következők: `antiword`, `atd`, `crond`, `ntpd`, `recode`, `rsync`, `screen`, `sudo`, `w3m`, `wvdial`, `xinetd`. A Commodore-emulátor mellett a másik tábor igényeit a magyar fejlesztésű ZX Spectrum emulátor elégítheti ki. Fejlesztők számára az `ltrace` és a `ccache` lehet érdekes, valamint természetesen az uhubuild rendszer, ami a számtalan hozzá tartozó segédprogrammal együtt ugyancsak megtalálható a terjesztésben.



Az irodai csomagok gyűjteményét a KJK-Kerszöv CD-Jogtárával, valamint a Lafisoft raktárkészlet-kezelő és számlázó programjával bővítettük.

A kikapcsolódni vágyóknak néhány új játékkal szolgálhatunk, ide tartozik például a Quadra nevű, több játékossal is játszható tetris, a bomberman típusú XBlast, a falkilövő Ibreakout, a GAV nevű röplabdajáték és a Jump n' Bump névre hallgató „nyuszi-őrület". Szórakozásképpen érdemes a figlet és a cowsay programokra is rápillantani.

G. T.: *Milyen változás tapasztalható a régi csomagok terén?*

K. E.: Az egyes csomagokban történt számtalan javításon kívül át is neveztünk, illetve több csomagot szétrobbantottunk. A pontos csomagnévváltozások listája megtalálható a lemezen.

A változatszám nélküli *.so* fájlok általában átkerültek a fejlesztői csomagba, csakúgy, mint a *.a* és *.la* fájlok, valamint a kézikönyv harmadik fejezetének az oldalai.

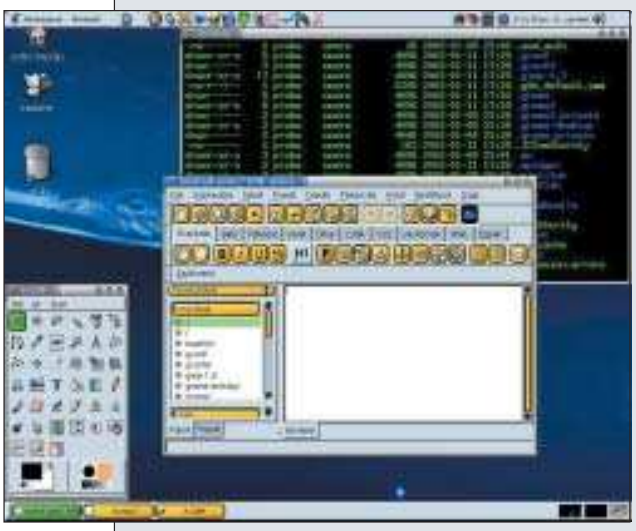
Az uhubuild rendszer következménye, hogy a terjesztés több téren is egységes lett, ezt mutatja például a csomagokról szóló leírásoknak a */usr/share/doc/Packages* könyvtárba történő gyűjtése, vagy a tömörítetlen és mindig rendszergazdai tulajdonú kézikönyvoldalak.

A rendszer egészét érintő változás, hogy a függvénytárakból a szimbólumok nincsenek kiszedve, azok az `nm` paranccsal megtekinthetők. Ez rendszerközeli fejlesztések során néha óriási segítség lehet.

G. T.: *Milyen változások történtek a magyar nyelv támogatásával kapcsolatban?*

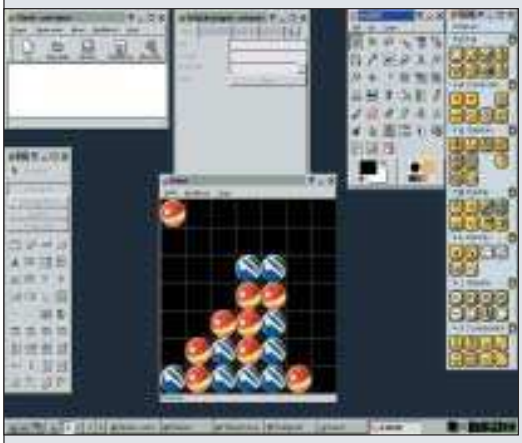
K. E.: Számtalan program magyar fordításán javítottunk, átfuttattuk helyesírás-ellenőrzőn, vagy éppenséggel a semmiből készítettük el. A rendszerben megtalálható

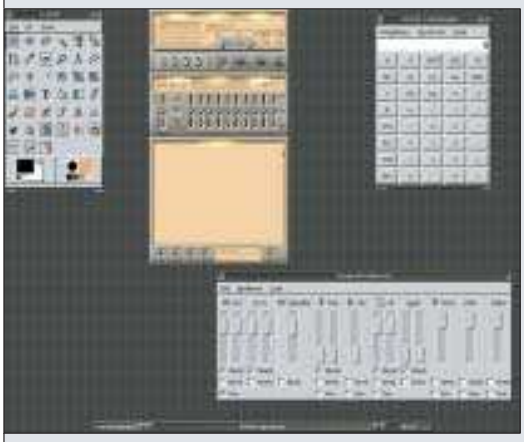
a Gnome magyar nyelvű súgója. Több Latin-2-es karakterkészletet szállítunk, így már az OpenOffice.org menüiben is szép ékezetek jelennek meg. Tucatnyi alkalmazást átállítottunk, hogy a Latin-2 karakterkészletet használja a Latin-1 helyett.
A glibc-ben a hárombetűs hónaprövidítéseket a magyar helyesírás szerinti alakra cseréltük ki (például „sze" helyett „szept").



A konzolt használók számára feltűnhet az ű betű ékezetének a kijavítása, valamint az, hogy a `gpm` az ő és ű betűket immár helyesen kezeli.
Természetesen a magyar `ispell` legújabb változatát szállítjuk, amely már az OpenOffice.org-ban használt komolyabb myspell motorhoz is tartalmaz parancssori felületet.
Az uhubuild rendszer még arra is figyel, hogy a csomagok leírásában lévő szavakat a magyar ispell elfogadja-e, így a leírások kapcsán minket ért sok kedvezőtlen bírálat remélhetőleg mostantól idejétmúlt lesz.
G. T.: *Gondolom több olyan változtatás is történt, ami akár egy önálló cikket is megérdemelne, de most csak néhány érdekesebb részre kérdeznék rá. Mi a helyzet a rendszerindítással?*
K. E.: Az LSB-hez igazodva a rendszer 5-ös futási szinten indul el. A 3-as szintre váltással leállítható a `gdm` vagy `kdm`. Hibajavítás az rc1-hez képest, hogy a démonok az `init` parancsfájl segítségével immár a csomag frissítése után is leállíthatók. Azok a démonok, amelyek csak a külvilág felé szolgáltatnak (`opensshd`, `proftpd`, `xinetd`), nem indulnak el alapból.
A rendszermag üzenetei nem kerülnek ki a konzolra, így a képernyő a rendszer indulása során áttekinthetőbb és szebb. A karakterkészlet és a billentyűzetkiosztás a legelső lépésben betöltődik, adatbázisuk */usr/lib/kbd* alól */etc/kbd* alá költözött. Ezt követi a fájlrendszer ellenőrzése, hogyha itt bármi hiba adódna, már be legyen töltve a kívánt billentyűzetkiosztás.
A */etc/sysconfig* alatt megadható, hogy a CMOS óra helyi vagy UTC időt tároljon-e, valamint hogy a NUM LOCK-ot ki- vagy bekapcsolt állapotban szeretnénk-e látni konzol alatt.
A felhasználónevet fogadó `agetty` programot kicseréltük az `fbgetty`-re, amely rendelkezik egy-két érdekes, szokatlan tulajdonsággal is, például ilyen a pontos idő kiírása és frissítése (igaz, ezt félig mi tettük bele). Továbbá új külsőt kapott az üdvözlő szöveg.
A `memtest` program immár a telepített csomagok között is megtalálható, jóllehet ez nem egy Linux alól futtatható program, viszont önműködően bekerül a GRUB indító menüjébe.
A `make-bootfloppy` parancs a telepített rendszer beállítófájljainak érintése nélkül készít indítólemezt – ez a szolgáltatás a telepítés során is jól működik.
Az indítólemezen az UHU-háttér előtt az előre beállított három lehetőség jelenik meg: a telepített rendszer indítása a merevlemezről szedett rendszermaggal és `initrd`-vel, a hajlékonylemezről szedett maggal (`initrd` nélkül), illetve a memóriaellenőrzés.
G. T.: *Mi történt a fájlrendszerrel?*
K. E.: A */usr/X11R6* alá csak az XFree86, illetve az ahhoz roppant közel álló csomagok pakolnak (karakterkészletek, valamint `lesstif`). Mindenki más a */usr/bin* alá szállítja a binárisokat, a */usr/lib* alá a libeket stb., még akkor is, ha az adott program X-et igényel.
A terjesztés részeként többé nem használjuk a */opt* könyvtárat. Az Acrobat Reader és az OpenOffice.org a */usr/lib* könyvtár alá költöztek át.
G. T.: *Sokan jelezték, hogy a grafikus terminálokon belül az egyes billentyűk hibás működése miatt gondjaik támadtak.*
K. E.: Ezeket remélhetőleg mind javítottuk, és most már mindenütt jól működik a HOME, END, PAGE UP, PAGE DOWN, BACKSPACE és a több.
A terminálkezeléshez kapcsolódik, hogy a Midnight Commander immár rendesen kezeli az egeret (a másolás–beillesztés szolgáltatás a SHIFT gomb lenyomásával érhető el), mindig színes, és rendszergazdaként a fájl törlésére vagy felülírására rákérdező ablak, valamint a hibaablakok piros alapon fehér helyett a még feltűnőbb sárga színt használják. Egyébként az `mc`-vel kiemelkedően sokat foglalkoztunk, melynek eredménye egy átdolgozott *mc.ini* és *mc.ext* fájl, sok hibajavítás, közben pezsgő kapcsolattartás a program fejlesztőjével. Ezek után talán nem meglepő, hogy a felhasználó nem a megszokott 4.5.55-ös változatával fog találkozni, hanem a 4.6.0-pre2-vel.
G. T.: *Változott-e valami a bejelentkezés, az azonosítás, a beállítófájlok körül?*
K. E.: A */etc/profile* és */etc/bashrc* egyszerűsödtek.
A `bash` önműködően hajtja végre a */etc/bashrc* fájlt, így a felhasználók saját könyvtára alatt többé nincs szükség arra a *.bashrc* fájlra, amelyik szinte mindig csak egy másolat a */etc/bashrc* fájlból. A PAGE UP és PAGE DOWN gombok a parancslistában a begépelt karakterekkel egyező kezdetű parancsok közötti előrefelé, illetve hátrafelé

történő keresésre lettek beállítva. A TAB gomb hatására a bash-completion csomag megpróbálja értelmes módon kiegészíteni a parancsokat. Általában egész jól sikerül neki. Ha valakinek nem tetszik, egy saját üres *~/.bash_completion* fájllal letiltható, de a csomag akár rendszergazdaként is leszedhető.
Új parancsjelet (prompt) terveztünk, amely egyrészt





már a gépnevet is tartalmazza (amennyiben az különbözik localhost-tól), másrészt az írásmódja olyan, hogy könnyen be lehessen másolni az `ssh` parancs tulajdonságai közé.
A környezeti változók legnagyobb része nem a */etc/profile*-ból állítódik be, hanem a PAM rendszer révén. Ezeket a változókat a */etc/security/pam_env.conf* fájlban, valamint a */etc/env.d* könyvtárban találjuk meg. Fontos megemlíteni, hogy a `su` parancs különböző PAM fájlt használ aszerint, hogy bejelentkezési héjat kell-e indítania, ugyanis ez esetben PAM-mal állíttatja be az új környezeti változókat. Újdonság a `pam_xauth` modul használata, amely akár `su`, akár `su -` esetén továbbítja az „X-sütiket”, így a grafikus felületen az új felhasználó is gond nélkül ablakot tud nyitni.
A grafikus felületen történő bejelentkezéssel kapcsolatban elsőre talán az tűnik fel a felhasználónak, hogy immár GDM-ben és KDM-ben is választhat a telepített ablakkezelők közül. Ezt is az `uhu-pkg` rendszerrel oldottuk meg.

G. T.: *Milyen újdonságról tudsz beszámolni az OpenOffice.org kapcsán?*

K. E.: A 046-os fordításra frissítettünk. A csomag a */usr/lib* könyvtár alá települ, változatszám-független könyvtárba. A felhasználó saját könyvtára alatt is változatszám-független könyvtárat hoz létre. Az indító parancsállomány sokat egyszerűsödött. Most már akkor is működik, ha a saját könyvtár nem */home/felhasználónév* alakú. A magyar helyesírás-ellenőrzése terén nem saját magával szállítja a szótárfájlokat, hanem a `myspell-hu` csomagra támaszkodik, így ez könnyebben frissíthető, kicserélhető alatta. A `fonts-urw-iso-8859-2` csomag tartalmaz olyan karakterkészleteket, amelyektől a magyar ékezetes betűk az *OO.o* menüiben is szépek lesznek. Ehhez az X beállítófájljában szerepelnie kell egy ilyen sornak:

```
FontPath
"/usr/X11R6/lib/X11/fonts/URW-latin2"
```

G. T.: *Az XFree86-ban is történt változás?*

K. E.: Az euró jelét felvettük a billentyűzetkiosztásba. Kiszedtük a félrevezető és elavult `l2_*` nevű változatot. Fekete háttérrel indul a jó öreg szokásos fekete-fehér pöttyös helyett, ahogyan majd a leendő 4.3-as változat is. Szinte teljesen újraírtuk az Xsession, xinitrc és startx parancsfájlokat. A már-már hagyományos *.Xresources*, *.Xmodmap* és *.Xkbmap* fájlokon túl *.Xnumlock* fájlt is támogatnak, ebbe egy „on” vagy „off” szót írhatunk, így belépésünk során a NUM LOCK a kívánt állapotra állítódik be.

G. T.: *Maradjunk ennek a témának a környékén. Mi történt a Gnome-val?*

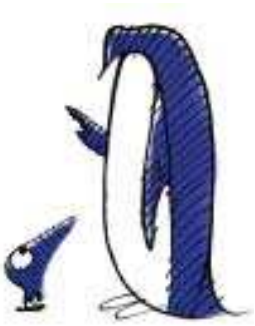
K. E.: Komolyabb változások ezen a téren nem történtek, bár ez leginkább annak köszönhető, hogy nem jelent meg lényegesen újabb Gnome-változat. Javítottunk egy-két fordítási hibát, így például a „Kezdd itt” egy d-vel írt változata, vagy az értelmetlen „Takarítás név alapján” immár a múlté. A rendszerfigyelőben is jók az ékezetek. Mivel saját billentyűzetkiosztást használ, az euró jelet itt is beállítottuk. A pingvinfej-ikon a `multi-gnome-terminal` helyett a jobb magyar fordítással rendelkező hagyományos `gnome-terminal`-t indítja.

G. T.: *A jelek szerint valóban nem túlzó az az állítás, hogy szinte minden megváltozott. Nyugodtan javasolhatjuk az áttérést az rc1-felhasználóknak?*

K. E.: Természetesen igen. Nagyon sokat fejlődött a terjesztés, és nem tudunk olyan tényezőről, amelyben az rc1-nél gyengébb volna.

G. T.: *Köszönöm a beszélgetést, és további jó munkát kívánok!*

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



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

A hónap szakmai tanácsai

A Snort, a MySQL és az ACID együttes használata

Tud valaki olyan leírást javasolni, amelyik teljes mélységében bemutatja a Snort telepítését egy olyan rendszerre, amelyen MySQL és ACID fut? Már három hete próbálkozom, de az ACID-konzol nem naplózza az eseményeket.
Colin Slevin, Colsev@transwareplc.com

Nem említetted, hogy a naplózás más módon megtörténik-e. A Snort általában több csatornán keresztül is naplóz, ha az alapbeállításait használod. Ha egyáltalán nem készül napló, valószínűleg máshol lesz a gond. Egyébként szerkeszd át a *snort.conf* állományt. Több naplózási példa is van ott, biztos találsz olyat, amelyik megfelel az igényeidnek. Van Snort-levelezőlista is, a felhasználók nagyon segítőkészek. Feliratkozhatsz rá a ➲ http://www.snort.org/lists.html oldalon. Javaslom, hogy ott tedd fel a kérdéseidet. Add meg pontosan, miképpen telepítetted a programokat, akkor részletesebb segítségre számíthatsz.
Chad Robinson, crobinson@rfgonline.com

A CD-s telepítés lefagy

Sokszor megpróbálkoztam a Red Hat 7.3 telepítésével egy NEC P150-es gépre. CD-ről indítom a rendszert, de a gép mindig lefagy a telepítés felénél.
Leong Y. C., leongyc@tp.edu.sg

Próbáld megváltoztatni a CD-ROM DMA-beállításait a BIOS-ban. Bizonyos gépek lefagynak, ha ez az érték nem megfelelő. Ez nemcsak a Linuxszal történhet meg, hanem bármely operációs rendszerrel, ami CD-ről futtat alkalmazást.
Felipe E. Barousse Boué, fbarousse@piensa.com

Három dolgot próbálhatsz meg:
1. alkatrészek lekérdezésének tiltását (rendszerindítás a `noprobe` kapcsolóval);
2. a szöveges módú telepítést;
3. ha megvetted a dobozos változatot, beszélj a Red Hat terméktámogatási szolgálatával.

Marc Merlin, marc_bts@google.com

A /boot bombabiztos másolata

Távolról üzemeltetek néhány Linux-munkaállomást, és nagyon fontos számomra, hogy mindig elérhetőek maradjanak. Úgy szeretném ezeket a rendszereket beállítani, hogy minden gépnek két betöltő és két gyökérlemezrésze legyen. A célom ezzel az, hogy az elsődlegesen használt lemezrészekkel kapcsolatos karbantartási feladatokat a másodlagos lemezrészekről indított rendszerrel végezhessem. A másodlagos lemezrészeken csak a legalapvetőbb képességekre van szükségem, név szerint az `e2fsprogs` csomagra, a DHCP-re és az `sshd`-re. LILO-t vagy Grubot tervezek használni, és az `init 6` paranccsal váltogatnék a két telepítés között. Feltéve, hogy a konzolos hozzáférés nem oldható meg, mi a legbombabiztosabb módja a lemezrészek kiosztásának és a beállításoknak egy ilyen rendszernél?
Marc Lavergne, mlavergne@cfl.rr.com

Nem igazán van szükséged másodlagos lemezrészekre a frissítések telepítéséhez. Az RPM lehetővé teszi a menet közben történő frissítést. Gyakorlatilag a rendszermag frissítése az egyetlen művelet, amely a gép újraindítását kívánja. Ezt észben tartva az általad vázolt oda-vissza megoldás jól jöhet akkor, ha a frissítésnél elszúrtál valamit. Tudtommal a LILO és a Grub nem rendelkezik azzal a képességgel, hogy egy másik lemezrészről indítsa a rendszert abban az esetben, ha az előző rendszerindítás sikertelen volt. Valószínűleg írnod kell egy pár kódsort ehhez a képességhez. Ez a megoldás sem véd meg azonban a rendszer lefagyásától vagy gép meghibásodásából adódó rendszerösszeomlástól. Valójában a távoli rendszerek is igényelnek bizonyos mértékű konzolos hozzáférést.
Christopher Wingert,cwingert@cwingert.qualcomm.com

Úgy látom, a megoldás nehézsége abban rejlik, hogy nem számíthatsz a távoli felhasználó segítségére – különben rendszerindító CD használatát javasoltam volna. Némelyik – például a Gentoo – SSH-val is rendelkezik.
Chad Robinson, crobinson@rfgonline.com

Nincs olyan lemezrész-elrendezés, amelyik jobb lenne, mint a többi. Az egyetlen dolog, amivel a rendszert bombabiztosabbá teheted, ha csak négy lemezrészt használsz, és nem használsz kiterjesztett lemezrészeket. Így a lemezrészeket leíró táblázat teljes egészében a merevlemez fő indítósávjában helyezkedik el, és nem terjed ki láncolt listaként a lemez távoli pontjaira. Ez természetesen nem jelent nagy különbséget, csak akkor, ha valami nagyon rossz dolog történik a merevlemezzel.
Marc Merlin, marc_bts@google.com

Linuxos illesztőprogram Belkin vezeték nélküli hálókártyához

Nemrég telepítettem egy vezeték nélküli PCI-hálókártyát, de a gyártó cég, a Belkin, nem adott mellé Linuxhoz való illesztőprogramot. Keresgéltem egy kicsit, és rátaláltam a Linux-WLAN illesztőprogramra. Megpróbáltam telepíteni, de a fordításkor (`make all`) rengeteg hibaüzenetet kaptam, és a program nem fordult le.
Matt Jacobs, mjacobs519@hotmail.com

Ha gyorsan szeretnél csatlakozni a hálózathoz, jó hírem van: a Belkin sok más gyártóhoz hasonlóan az elterjedt Prism/2 lapkakészletet használja. *Tim Miller* készített egy Prism/2 alapú kártyákat támogató RPM-et a Red Hat 7.3 és 8.0 terjesztésekhez (➲ http://prism2.unixguru.raleigh.nc.us). Forrásból szeretnéd lefordítani a meghajtókat, akár azért, mert egyéni rendszermagot szeretnél, akár azért, mert meg akarod tanulni, hogyan kell fordítani az ilyesmit. Ehhez olvasd el a rendszermag HOGYAN-okat (➲ http://www.linux-hogyan.hu/indexhu.php?page=howto/translations/hu/html_single/Kernel-HOWTO-hu.html).
Don Marti, dmarti@ssc.com

Új termékek

Runtime Revolution 2.0

A Runtime Revolution bejelentette a Revolution 2.0, egy több felületen használható fejlesztőeszköz megjelenését. A Revolution felületépítőt, egy, az angolhoz hasonló programnyelvet, valamint beépített internetes és multimédiás lehetőségeket használ. A felhasználó által megírt egyetlen alkalmazás minden célopercációs rendszeren futni fog. A 2.0-s változat újdonságai közé sorolható: Jaguar-barát felület, unicode-os szövegbevitel és szövegműveletek, számolótábla, illetve táblázat-szövegmezők, új XML-programkönyvtár, új SOAP-támogatás, új jelentésnyomtató motor, beépített programozható hibakereső, továbbfejlesztett adatbázis-elérés minden felületen, Perl-szerű szabványos kifejezések és eszközök MIDI-zenefájlok létrehozásához és lejátszásához.
Adatok: Runtime Revolution Ltd., 91 Hanover Street, Edinburgh, Scotland, UK, EH2 1DJ, telefon: 970-278-9243, e-mail: support@runrev.com,
➲ http://www.runrev.com/newsrevolution20.html

Nuxeo Collaborative Portal Server

A Nuxeo Collaborative Portal Server (CPS) egy Zope-ra épülő webes tartalomkezelő rendszer. A CPS használatával csoportmunkát segítő intranetes, extranetes vagy internetes alkalmazások készíthetők. A CPS-ben a weboldalak és a rendszerszerkezetek létrehozása dinamikusan történik. A tartalom, az adatok megjelenítése és az adatokon végzett műveletek el vannak választva. Számos formátumban (StarOffice, OpenOffice, PDF, RTF, XML stb.) lehetséges a többféle formátumú dokumentumok indexelése és előnézet megjelenítése. Beépített keresőszolgáltatást, metakönyvtárat, változatkezelést és levelezőlistát is tartalmaz. A CPS szabad program, a Nuxeo webhelyéről tölthető le.
Adatok: Nuxeo 14, rue du Soleillet, Paris, France 25020, e-mail: contact@nuxeo.com,
➲ http://www.nuxeo.com/en

7160 Anything I/O

A 7160 az FPGA alapú Anything be/kimenetvezérlők önálló (nem sínes) változata. Az eszköz 96 B/K biten dolgozik, csatlakozónként 24 bittel négy ötventűs csatlakozón. A csatlakozók tűkiosztása megfelel a B/K-modulszekrényeknek. A 7160 rendelkezik még két RS-485 soros kapuval és két RS-232 kapuval. Ezek közül az egyik a kezdeti beállítások letöltésére szolgál a kártyára épített Flash EEPROM-ba. Számos B/K-beállítóállomány rendelkezésre áll, többek között egyszerű és okos távoli B/K, négy- és nyolctengelyes robotmotor-vezérlő stb. Kiegészítő kártyák is kaphatók, például nyolccsatornás RS-232/485 csatolófelület, hibakereső LED-kártya, kéttengelyes #A léptetőmotor-meghajtó.
Adatok: MESA Electronics, 4175 Lakeside Drive, Suite 100, Richmond, California 94806, e-mail: sales@mesanet.com,
➲ http://www.mesanet.com

QuickStart Kitek

Az Ampro Computers és a TimeSys összefogott, hogy tanúsított, ipari minőségű Linux-terjesztéseket és fejlesztőkörnyezeteket hozzanak létre a beágyazott valós idejű OEM-eszközök tervezéséhez és telepítéséhez. Az első közös termék a QuickStart Kit (QSK), amely az Ampro PowerPC ISA alapú EnCore PP1 moduljához való. A MIPS és x86 alapú kártyákhoz és modulokhoz is hamarosan megjelenik a megfelelő QSK. Minden QuickStart Kit tartalmazza a TimeSys Linux terjesztését, melynek része az operációs rendszer, az eszközmeghajtók, a GNU fejlesztőeszközök, fejlesztőkörnyezetek Windows (Cygwin) és Linux munkaállomáshoz, a gyökérfájlrendszer és az ellenőrzött rendszerbetöltő program. Ráadásul a programcsomag néhány betölthető rendszermagmodul kipróbálásra szánt változatát is tartalmazza, amelyek a valós idejű képességeket adják a TimeSys Linux RealTime, TimeSys Linux CPU, illetve a TimeSys Net rendszerekhez.
Adatok: Ampro Computers, Inc., 5215 Hellyer Avenue #110, San Jose, California 95138,
➲ http://www.ampro.com/linux

Xandros Desktop

A Xandros kiadta a Xandros Desktop nevű munkaasztali környezetet, amelyet a Corel Linux Business részlegének csapata kezdett el fejleszteni, még mielőtt a Xandros 2001-ben felvásárolta volna őket. A Xandros legfőbb tulajdonságai: cseresza-batosság, egyszerű kezelés, nagy biztonság és a hálózaton keresztüli frissítés lehetősége. A Xandros képes együtt dolgozni Windowst, Unixot és Linuxot futtató hálózatokkal, ezek között a rendszerek között az állományok megosztása is lehetséges.
A munkaasztali környezet része a CodeWeaver CrossOver megoldása, amelynek segítségével a felhasználók egy időben képesek Windows- és Linux-alkalmazásokat használni. Az öt lépésből álló grafikus telepítőprogram az NTFS-lemezrészek átméretezését is lehetővé teszi.
Adatok: Xandros, Inc., 41 East 11. Street, 11. floor, New York, New York 10003, e-mail: sales@xandros.com,
➲ http://www.xandros.com

SuSE Openexchange Server

A SuSE Linux bejelentette az Openexchange Servert, egy teljes körű kapcsolattartási és csoportmunka-megoldást. Az Openexchange egy alkalmazáson belül kínál elektronikus levelezést, a webes levelezőügyfelek támogatását, központi találkozó- és címjegyzékkezelést, valamint projektirányító és feladattervező eszközöket. A központilag ellenőrzött dokumentumkezelés és a csoportalapú vitafórumok szintén rendelkezésre állnak. Az Openexchange felület a böngészőprogramon belül fut, így a szolgáltatások minden számítógéptípuson és operációs rendszeren használhatók, beleértve a Windowst és a Mac OS-t is. Az adatok egyeztetése kézi számítógépekkel (PDA) is lehetséges.
Adatok: SuSE Linux AG magyarországi irodája, 1399 Budapest, Pf: 701, 301
Tel: (1)450 0921
Fax: (1)330 5672 e-mail: info@suselinux.hu
Anyacég: ➲ http://www.suse.de/en

Linux Journal 2003. január

diff -u: Rendszermag-fejleszési hírek

Jeff Dike User Mode Linuxát végre-valahára bevették a 2.5-ös rendszermagfába. Így a felhasználók a Linux-rendszermagot a rendszerben futó felhasználói folyamatként hívhatják meg. Másként fogalmazva: párhuzamosan indítható tetszőleges számú virtuális Linux-rendszer, amelyeket Beowulffal vagy más módon telepbe lehet rendezni; illetve olyan új vezérlőprogramok vagy egyéb gyorsan terjedő javítócsomagok kipróbálására használhatók, amelyek a rendszer összeomlását okozhatják. Nincs több hosszas újraindítás, ha az új játékszer lefagy – indíts új példányt, és folytasd a kísérletezést!

Az SGI naplózó fájlrendszere, az XFS is bekerült a 2.5-ös fába, miután végre eleget tett *Linus Torvalds* elvárásainak. Az XFS sokáig meglehetősen vad változtatásokat hajtott végre a rendszermag egyes területein, amelyeket Linus véleménye szerint nem lenne szabad érinteni. Ezeket a „kinövéseket" lenyesték, a kód pedig bekerült a fába. A naplózó fájlrendszerre vágyó felhasználók az ext3, a ReiserFS, az IBM-féle JFS és – most már – az XFS közül választhatnak.

A számlakezelés nagy változáson ment keresztül: most százezer szál futtatható párhuzamosan. Ez megdöbbentette a rendszermagfejlesztőket, Linus például azt hitte, félreolvasta a bejelentést. *Ulrich Drepper* és *Molnár Ingó* kardoskodott ezért a változtatásért, amellyel a szálméretezés messze meghaladja azt az ésszerű határt, ameddig az otthoni rendszereken el lehet merészkedni. Másrészt viszont – ahogy az egyik fejlesztő mondta – ha semmi másból nem, ebből biztosan látszik, hogy jót is cselekedtünk.

Folyamatban van egy olyan kezdeményezés, amely gyakorlatilag minden felhasználót komolyan érinteni fog. Előfordulhat, hogy visszavonják azt a lehetőséget, hogy egy modult a rendszermagból ki lehessen üríteni (unload). Úgy tűnik, a modulkezelés kódja túl összetetté vált, és az egyik egyszerűsítő javaslat szerint a már betöltött modulokat tartósan be kellene illeszteni a futó rendszerbe. Csak az a gond, hogy a modulkiürítést sok fejlesztő hasznosnak tartja és nem akar lemondani róla. A jelenlegi állás szerint a vita bármelyik irányba eldőlhet.

Új összeomlás-kezelési kódok láttak napvilágot. Ezek egyike egy olyan javítócsomag, amely a memóriakiíratási (core dump) fájlok nevét szabályozza. Amikor egy programhiba miatt memóriakiíratásra kerül sor, akkor a pillanatnyi könyvtárban nem egyszerűen egy core nevű fájl jön létre, hanem a fájl a felhasználó által megadott néven egy másik könyvtárban is létrehozható. Az összeomlás kezelésére szolgáló másik kódrészlet a `kksymoops`, egy OOPS-kezelő, ami feloldja a rendszermagszimbólumokat, még mielőtt kiíratná az OOPS-adatokat. Korábban az OOPS-kimenetet külön programmal kellett feldolgozni, hogy a rendszermagfejlesztők használni tudják. A `kksymoops` segítségével több és hasznosabb adat válik elérhetővé. A felhasználók számára egyszerűbbé válik a rendszerösszeomlás bejelentése, a fejlesztők számára pedig a hibakeresés.

➲ http://www.kernel.org
➲ http://user-mode-linux.sourceforge.net

Zack Brown

Linux Journal 2003. január, 105. szám

Falatkák

CheckBook Tracker

Ha le szeretnéd cserélni a Quickent, eljött az idő. Én nem használtam a Quickent, ezért QIF formátumban letöltöttem a banki fájlokat, és gond nélkül felvettem őket a CheckBook Tracker programba. A CheckBook kezelőfelülete áttekinthető, hatékony és többféle megjelenítési módot kínál. Talán a legegyszerűbben használható csekkönyv, amivel az utóbbi időben dolgom volt. Futtatásához szükségesek: libXi, libXext, libX11, libm, libgtk, libgdk, libglib, libgdk_pixbuf, libgmodule, libdl és, glibc.
➲ http://tony.maro.net

TypeFast

A TypeFast egyszerű, leckékre bontott gépírásgyakorló program. Nem tanít meg gépelni (sőt feltételezi a gépírás ismeretét), de a segítségével gyorsabban leszel képes gépelni `xterm` vagy VT terminálablakban. Különféle billentyűzeteken gyakorolhatsz (QWERTY vagy DVORAK típusún), és kezet is válthatsz, hogy mindkettőt fejleszthesd. Futtatásához libncurses és glibc szükséges.
➲ http://homepages.ihug.co.nz/~syringe/typefast.html

David A. Bandel

Linux Journal 2003. január, 105. szám

A nyelvtörvényről – utólag

A 2001-es, röviden csak *nyelvtörvényként* emlegetett szabályozás elfogadása hosszadalmas és nehéz, máig sem lezárt társadalmi–szakmai vita egyik állomásának volt tekinthető. Az alábbiakban a nyelvtörvényen túl röviden érintjük a vita lényegi pontjait, tágabb hatósugarát, valamint az egyes táborok főbb érveit.

> „Magyar nyelven kiadott sajtótermékben, magyar nyelvű rádió- illetve televízióműsorban, továbbá szabadtéri reklámhordozón közzétett, a gazdasági reklámtevékenységről szóló 1997. évi LVIII. törvényben… meghatározott gazdasági reklámban a reklám szövegét, ideértve a jelmondatot (szlogent) is – a vállalkozás neve, megjelölése, illetve az árujelző kivételével – magyar nyelven meg kell jeleníteni, függetlenül a közzététel módjától. A... követelmény teljesíthető úgy is, hogy ugyanazon reklámban az idegen nyelvű szövegen (szövegrészen) kívül annak magyar nyelvű megfelelőjét is megjelenítik, legalább ugyanolyan jól érzékelhetően, valamint legalább ugyanolyan méretben, mint az idegen nyelvű szöveget (szövegrészt)".
>
> *(Részlet a 2001. évi XCVI. törvényből)*

1997 júniusában a Parlament elfogadott egy reklámtörvényt. Ennek kapcsán egyes nyelvvédő civil szervezetek a reklámok nyelvezetének „túlzott idegenszó-használata" (*Grétsy László*) miatt már akkor azt szorgalmazták, hogy a törvény az idegen nyelvű reklámok megjelenését is korlátozza, mivel azok személyiségi jogokat csorbítanak.
A későbbi, 2001-es törvényjavaslatot érintő kormányzati érvelés szerint is hazánkban minden polgárt alkotmányosan megillet az „információs szabadságjog" (*Hende Csaba*, az Igazságügyi Minisztérium volt politikai államtitkára, egyúttal a törvény szövegének írója), vagyis hogy ne csak lássa, de értse is például az üzletek nem magyar nyelvű feliratait. Az akkori törvényjavaslat ellen félszáz magyar nyelvész, köztük akadémikusok is, nyilatkozatban tiltakoztak (lásd Élet és Irodalom, 1997. 05. 09.), és jelentős vita bontakozott ki. Később, 2001 márciusában az addigra elkészült T/4899. számú törvényjavaslat (a magyar nyelvnek a gazdasági reklámok és az üzletfeliratok, továbbá egyes közérdekű közlemények közzététele során való használatáról) kapcsán ez a vita újjáéledt – először a Népszabadság hasábjain (*Grétsy László* és *Kálmán László* nyelvész között), majd más fórumokon is (csak a sajtóban több mint harminc cikk és olvasói levél jelent meg, megszólalt többek között nyelvész, szociológus, író–költő és politikus is).
Voltak olyanok, akik a törvényjavaslatban „a reklámnyelv szabályozása mögé bújtatott nyelvtörvényt" (*Kis Tamás*) láttak, mások kiemelkedő fontosságú nyelvvédő szabályozást. A két tábor vitája azonban egy nagyobb szakmai, azaz nyelvészek között zajló vitába ágyazódott be, ez pedig nem volt más, mint a nyelvművelők és a nyelvészetből a nyelvművelést kizáró nyelvészek között ma is zajló csatározás (az ellentét elsősorban a nyelv előíró vagy leíró szemlélete melletti állásfoglalás különbségében gyökerezik). A később elfogadott nyelvtörvényt szorgalmazók jórészt az „anyanyelvész" (*Deme László*) nyelvművelők soraiból kerültek ki, míg a velük szemben-állók az „általános és elméleti nyelvésznek" nevezett, magukat többnyire szociolingvistaként meghatározó szakemberek közül. Az előbbiek úgy vélték, hogy a legutóbbi évtizedekben a magyar nyelv az őt érő idegen nyelvi hatásokkal nem vagy csak nehezen képes megbirkózni, ezért azokat ellensúlyozni kell (ugyanakkor a magyarítás a meghonosodott szavakat már nem érintené). Az utóbbiak azonban úgy látták, hogy a törvényjavaslatot nem előzte meg tudományos vita és reprezentatív felmérés, továbbá a javaslatban foglaltak nyelvészeti szempontból nem megalapozottak. Megítélésük szerint a törvény bújtatott, mivel bármilyen kis területen is, de a magyar *nyelvhasználatot* szabályozná (lásd a törvényjavaslat címét), ami voltaképpen nyelvi hátrányos megkülönböztetés. A törvény ugyanakkor egy, az MTA elnöke által kijelölt tanácsadó testület irányadó állásfoglalására hivatkozik, mint olyanra, amely a törvényjavaslat elfogadása után a kérdéses nyelvi kifejezések meghonosodottságának megítélésében és nyelvhelyességi kérdésekben illetékes lesz…

KAPCSOLÓDÓ CÍMEK

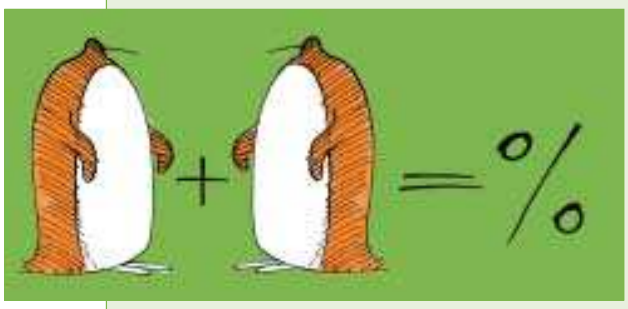
A 2001. évi XCVI. törvény szövege
➲ http://www.complex.hu/kzldat/t0100096.htm/mun_2.htm
Országgyűlési vita a nyelvtörvényről
➲ http://www.mkdsz.hu/n13/parl010903nyelvtorveny.html
Készült egy, a nyelvtörvénnyel kapcsolatos írásokat összefogó oldal is, amelyen az összes cikk, hozzászólás elérhető
➲ http://delfin.klte.hu/~tkis/

A reklámszakma részéről a Magyar Reklámszövetség elnöke, *Levendel Ádám* a megoldást nem a megszorító intézkedésekben látta volna, hanem a jutalmazásban. Javaslata szerint a reklámkészítők között divattá kellene tenni a magyar nyelv használatát: szövegíró versenyeket lehetne hirdetni és a legjobb megoldásokat jutalmazni. Más hozzászólók szerint is a szakmára, illetve az egyes üzletek tulajdonosaira érdemes bízni a választást, mivel a szabályozás a magánszférát érinti.
A törvényjavaslatot a 2001. októberi országgyűlési vita után a parlament november 27-én végül elfogadta. A törvény előírta követelmények teljesülését a Fogyasztóvédelmi Főfelügyelőség és az üzlet működési engedélyét kiadó jegyző ellenőrzi. A büntetés mértéke 100 000 forintig terjedhet, illetve az üzlet bezárását vonhatja maga után.

Dér Csilla (dercsilla@freemail.hu)
A Linuxvilág olvasószerkesztője, fiatal kutatóként dolgozik az MTA Nyelvtudományi Intézetében.

Linux-index

1. Ennyi linuxos Sicom Systems „eladási pont” rendszert telepítettek a Puerto Ricó-i Burger King éttermekben: 160
2. Linux alapú PDA-k száma: 16
3. Ennyi milliárd keresést hajtanak végre havonta a Linux alapú Google keresőprogrammal: 5
4. A Linux-felhasználók száma: 18 millió



5. Ennyiedik helyen áll a Linux Professional Institute (LPI) minősítés a leggyorsabban terjedő minősítések listáján a Computer Reseller News (CRN) szerint: 1
6. Ennyiedik helyen áll a Red Hat Certified Engineer (RHCE) minősítés a leggyorsabban terjedő minősítések listáján a Computer Reseller News szerint: 2
7. A kiskereskedelmi Red Hat Linuxot vásárlók ennyi százaléka használja az operációs rendszert otthon: 34
8. A kiskereskedelmi Red Hat Linuxot vásárlók ennyi százaléka használja az operációs rendszert a munkahelyén: 13
9. A kiskereskedelmi Red Hat Linuxot vásárlók ennyi százaléka használja az operációs rendszert mind otthon, mind a munkahelyén: 50
10. *Hilary Rosen*-nek, a RIAA elnökének 2001. évi fizetése: 1 163 729 dollár
11. *Jack Valenti*-nek, az MPAA elnökének 2001. évi fizetése: 1 030 000 dollár
12. Átlagosan ennyi kenőpénzt fizetnek a hanglemezkészítő vállalatok az amerikai kereskedelmi rádióknak azért, hogy egy dalt felvegyenek a dallistájukra: 1000 dollár
13. Ennyi kenőpénzt fizet az amerikai hanglemezkészítő ipar hetente: 300 000 dollár
14. Az amerikai rádióadók ekkora összeggel támogatják a hanglemezkészítő vállalatokat vagy a művészeket, amiért azok zenét játszanak: 0
15. A kis internetes rádiók körülbelül ennyi százalék adót fognak fizetni a zenészeknek a RIAA behajtási szerve révén: 8–12
16. A tévéreklámok ennyi százalékát nem nézik azokban a háztartásokban, ahol linuxos TiVót használnak: 88
17. Húszévenként ennyied részére csökken egyetlen bit feldolgozási költsége: 1000

Források

1–2.: LinuxDevices.com
3.: Blogads
4.: Linux Counter
5–6.: Computer Reseller News
7–9.: InternetNews
10–11.: Washington Business Forward
12–13.: Salon
14–15.: MediaPost
16–17.: Martha Rogers, Peppers & Rogers Group

Linux Journal 2003. január, 105. szám

(Linux + Java) – Bul = Kaii

A Sharp Zaurus mellett az Infomart Kaii is megjelent a linuxos tenyérgépek között. A Zaurushoz hasonlóan a Kaiin is a Lineo Embedix Plus Linux 2.4.2-es változatú rendszermagja, Trolltech Qt/Embedded és Qtopia környezet, valamint Insignia Java Virtual Machine fog futni. A Kaii processzora azonban (a Zaurus StrongARM processzorával szemben) egy 160 MHz-es Hitachi SH3-mal és képernyős billentyűzettel rendelkezik.

A Kaii színes vagy monokróm képernyővel (felbontása 320×240) kapható, legfeljebb 128 MB RAM-mal rendelkezhet, továbbá 32 MB ROM-ot vagy flashmemóriát, USB-csatlakozót, soros RS-232C és IrDA kaput, CompactFlash Type II és MMC kártyahelyet tartalmaz. Rajtuk kívül megtalálható rajta az Opera böngésző, a Hancom Mobile Office programcsomag, a `sync` program, ami a linuxos asztali gépekkel, az OS X és Windows-rendszerekkel biztosítja az együttműködést, illetve egy az Outlookhoz hasonló személyes adatkezelő program is futtatható rajta. Mérete 137×73×17 mm, vagyis csaknem ugyanakkora, mint a Zaurus zárt billentyűzettel. Az Infomart szerint a Kaii „költségtakarékos számítógép-megoldás vagy különleges célra használt vállalati munkaállomás költségérzékeny piacok számára, mint amilyen India, Kína, Kelet-Európa és Afrika”. Az Infomart központja Indiában található.

Doc Searls

Linux Journal 2003. január, 105. szám

Még néhány szó a nemzetköziségről

Vajon alkalmas-e bármelyik kormány arra, hogy rábízzuk a műszaki fejlődés ösztönzését?

2002. novemberi lapszámunk témája a nyelvi nehézségek áthidalása volt, és több érdekes cikket is olvashattunk. Különösen megragadott *Wayne Marshall* „Elektronikus levelezés rádióval Nyugat-Afrikában" című írása, amely megmutatta, mire képes az ember, ha hajlandó kitágítani a látókörét, és felhasználja a rendelkezésére álló eszközöket – Wayne esetében nagy hatótávú, de kis sávszélességet nyújtó rövidhullámú rádiókról volt szó. Sajnos a nyomtatott lap szűkös terjedelme miatt a cikkben ismertetett összeállítás néhány részletét el kellett hallgatnunk, a teljes változatot azonban a ➲ http://www.linuxjournal.com/article/6299 címen elérhetővé tettük. Visszatérve a nyelvi gondokra, *Fred Noronha* érdekes Linuxszal és nyílt forrású fejlesztésekkel kapcsolatos híreket szállított Indiából és a környező országokból. A „köznép" lelkesen támogatja a szabad programokat, és a kormányzat is érdeklődik az állami rendszerek nyílt megoldásokra való átállítása iránt. Vagy mégsem? Következő webes írásában – ➲ http://www.linuxjournal.com/article/6389) Fred azzal riogat, hogy az indiai kormány valójában csak felhasználja a szabad programokat, hogy a Microsofttól alacsonyabb árakat vagy helyi beruházásokat csikarjon ki. Az olvasók egyike szerint India nem bízhatja a kormányra a szabad megoldások sorsát. „Ha az emberek szabad programokat fognak használni, a kormánynak nem marad más lehetősége, követnie kell őket."
Végül – bár nem állt szándékunkban – sikerült némi zűrzavart keltenünk. Egy sajtóközlemény formájában november közepén ártatlan módon közzétettük weboldalunkon a 2002 Reader's Choice Awards nyertesét (➲ http://www.linuxjournal.com/article/6380). Nem hittük volna, hogy ilyen sok hozzászólás fog érkezni, eddig a pillanatig egészen pontosan 42. Kik is szólnak hozzá egy sajtóközleményhez? Hát azok, akik egyvalamiben értenek egyet: szavazóink, olvasóink bolondok, mert *rossz* nyerteseket választottak. Néha úgy érzem, hogy újra a homokozóban vagyunk, és kis piros dömperekkel püföljük egymás fejét, ha valami nem tetszik. Tessék tehát ellátogatni weboldalunkra, és nyelni egy kis homokot. Ha másokkal is meg szeretnéd osztani egyedi rendszered leírását, legyen az akár otthon, akár távol, írj pár szót az ljeditor@ssc.com címre. A Linux Journal oldalát megéri látogatni, naponta új írások jelennek meg rajta.

Linux Journal 2003. január, 105. szám

Heather Mead
A Linux Journal társszerkesztője. Szabadidejében fotózik, filmeket néz, zenét hallgat, valamint novellákat ír.

Rendezvénynaptár 2003

Folytatjuk Rendezvénynaptár sorozatunkat, és immár több magyar megmozdulásról hírt tudunk adni (ha valaki tud egyéb rendezvényekről, kérjük, írja meg nekünk!)

OpenOffice.org maratoni hétvége
Időpont: 2003. február 7–9. (Lásd a leírást a 36. oldalon.)

PHP Konferencia
Időpont: 2003. március 29.
Helyszín: ELTE Radnóti Miklós Gyakorlóiskola
Cím: 1146 Budapest, Cházár A. u. 10.

09.00–10.00	Regisztráció
10.00–10.20	Megnyitó
10.20–10.40	*Hojtsy Gábor (Goba)* Bevezetés a PHP világába
10.40–11.30	*Bártházi András* Portálok kialakítása, felépítése, biztonsági követelmények
11.30–11.40	Szünet
11.40–12.10	*Szabó Dénes* A Smarty sablonrendszer bemutatása
12.10–12.40	*Palócz István* ELTE radnóti Miklós Gyakorlóiskola PHP-oktatási tapasztalatok
13.00–13.30	Előadás
13.30–14.30	A PHP-szakosztály alakuló ülése
14.30–14.50	Zárás

A konferencia ingyenes, csak feliratkozni kell!
➲ http://www.radnoti.hu/~pp/phpconf/tamogatok.php

IT 2003 Konferencia és Kiállítás
Időpont: 2003. április 9–10.
Helyszín: MTA Székház
Budapest, V. ker., Roosevelt tér 9.
Konferenciairoda: 391–5889

Comdex Hungary
Információtechnológiai és telekommunikációs szakvásár
Időpont: 2003. április 16–19.
Helyszín: SAP Rendezvénycsarnok
1146 Budapest, Ifjúság útja 2.
Elérhetőség: Syma Expo Kft., 239-6120

X. Budapesti Nemzetközi Könyvfesztivál
Időpont: 2003. április 24–27.
Helyszín: Budapest Kongresszusi Központ
1123 Budapest, Jagelló út 1–3.
A Fesztiváliroda elérhetősége: 343-2537

Nyílt forrás az üzletben?

Nagyon sokszor írunk nyílt forrású fejlesztésekről – a Nyugatról, hogy ott menynyire hatékonyan oldanak meg komoly feladatokat nyílt eszközökkel, a fejlett országok minisztériumairól, ahol – gyakran biztonsági okokra hivatkozva – a nyílt rendszereket részesítik előnyben, akár a tengeren innen, akár túl. Mégis úgy érzem, idehaza még mindig megdöbbentően kevesen gondolkodnak „nyíltan". Tényleg nincs létjogosultsága ennek a világszerte ünnepelt és használt elméletnek kis hazánkban?

Pontosítsuk egy kicsit, mire gondolok. Ahogy viccesen mondani szokták, kétféle ember van: az egyik, aki mindent két külön csoportba oszt, és a többiek. Komoly ellentét alakult ki a nyílt és a zárt rendszerek pártolói között, már-már kívülről tudjuk mindkét oldal először felhozott érveit. A zárt programok követői szerint például: a zárt rendszer egy kézben összefogott fejlesztés, ha a cég vásárol egy könyvelőprogramot, elvárja, hogy a program ilyen és ilyen szabályoknak feleljen meg, megfelelő leírást, támogatást vár, ha nem, visszakövetelheti az árát, pereskedhet. A zárt rendszer kódjába nem lát bele senki, a hozzáértők készítik el, és nem is kívánatos, hogy a felhasználók belenyúljanak.

A zárt rendszerhez a követést az eladó készíti, egy-egy termék mögött gyakran egy egész vállalat áll.

Természetesen a nyílt oldal is felsorolja gyorsan a legtöbb említett témát, de pont a másik oldalról megvilágítva: a zárt rendszer kulcsa egyetlen kézben van, így a vásárló kiszolgáltatja magát a programozónak. Ha a rendszer nem tud valamit, és a beszállító tojik a fejére, a vásárló futhat a kifizetett nagy kalap pénze után, akár pereskedhet is. A zárt rendszer kódjába nem lát bele senki – lehet, hogy akik készítették, beleírtak még pár sort, hogy minden hónap tizedikén biztosítva legyen a havi fizetésük. Netán következetesen néhány hibát is ejtettek, hogy a vevőnek a követés megvásárlására is legyen oka. A zárt rendszert készítő cég közvetlen érdeke, hogy sokat keressen, és nem az, hogy a felhasználó minél gyorsabban, kényelmesebben tudjon dolgozni a termékkel. És így tovább…

A minap például egy cégvezetővel beszéltem, aki hosszú évek óta készít komoly programrendszereket mind windowsos, mind linuxos környezetben. Beszélgetésünk során szóba került ez a kérdés is, hogy vajon neki milyen tapasztalatai vannak a nyílt fejlesztésekkel kapcsolatban. Az egyik legérdekesebb tapasztalatot, amit megosztott velem, el is neveztem kosarasnak: a „nagyhalak" világában egy nagyvállalat számára ajánlott rendszerek közül az igazgató többnyire nem szakmai szempont szerint választ. Sőt gyakran nem is a hozzáértő választ az ajánlatok közül, hanem az igazgató „egységsugarú D-kosaras titkárnője". Más szavakkal megfogalmazva, nem biztos, hogy a program használhatósága a fontos, sokkal inkább az, hogyan adjuk el. És itt van az egyik legnagyobb különbség a két tábor között.

Nyílt rendszert árusítani?

A Debian és az RPM alapú Linux-változatok közti egyik legnagyobb különbség, hogy a Debian mögött nem áll egy óriáscég, amelyik – sok más tevékenysége mellett – erős reklámtevékenységet folytat, tárgyal, kiajánl, megken, rábeszél, elad. De ha összehasonlítjuk a linuxos cégeket a piacon zsíros falatokat lenyelő cégekkel, gyakran egy alapvető különbséggel találkozunk: a nyílt megoldásokat szállító cégek nem abból akarnak megélni, hogy kijárnak a vevőhöz, újratelepítgetnek, majd a vastagabbik ceruzájukkal megírják a számlát. Egy hozzáértő szakember egyszer üzembe helyez egy gépet, és utána néha rászagol, bírják-e még az alaplap kondenzátorai.

Meg lehet akkor élni a nyílt rendszerekből? Nézőpont kérdése: attól függ, mit tekintünk célnak. Ha a cél az, hogy minél nagyobb összegeket kérjünk el ugyanazért a termékért, akkor a nyílt fejlesztésekben szereplők éhhalálra vannak ítélve. Ebből a szempontból arany életük van például a könyvelési rendszereket készítő és forgalmazó cégeknek: a tíz éve megalkotott, azóta már relikviának számító ősállat-eszközökkel megírt, mondjuk paradox táblákon dolgozó könyvelőprogram azóta is működik, néha-néha módosítgatunk rajta itt-ott, ezt eladjuk párszáz felhasználónak, majd évente kiszabunk egy arcátlan követési díjat. De természetesen külön díjat szabhatunk ki például azért, ha valaki több tételt akar felvinni. Igaz, a rendszernek hihetetlenül mindegy, hogy tízezer vagy kétszázötvenezer tétel kerül feldolgozásra, de valamiből meg kell élni, nem igaz? Ezt a folyamatos bevételt biztosító elméletet irigyelte meg sok óriáscég is, akik már évek óta azon küzdenek, hogy valami fondorlatos módszerrel mindenki folyamatosan köteles legyen nekik nagy pénzeket fizetni a – felhasználók oldaláról nézve azért a szolgáltatásért, amit egyszer már megvásároltak. Nézzük, hogyan épül fel a nyílt megoldásokat szállító cégek világa. A vállalatnál felmerül az igény egy új vezérlőprogramra, ami már többcsapágyas gumipitypang-ültetvény locsolórendszerét is EU-szabványoknak megfelelően vezérli. A cégnek több lehetősége is van:

a) elmegy a boltba, és vásárol egy dobozos locsolóvezérlő programot;
b) egy céget bíz meg a fejlesztéssel; illetve
c) felvesz néhány programozót belső program fejlesztésére.

Melyikkel jár a legjobban? Az a) változat esetén kérdés, hogy a dobozos termék EU-megfelelő program-e. Ha nem, akkor pórul járt. Esetleg visszaveszekedheti a pénzét, vagy reménytelen kísérletet tehet a gyártónál arra, hogy újraírják a programot, hogy az ő ültetvényén is használható legyen. Ez a megoldás tehát csak akkor működik, ha a program minden feladatot ellát, és hiba sem merül fel. Erre pedig vajmi kevés remény van.

A második lehetőség lényegében még komolyabb kiszolgáltatottságot jelent, hiszen a cég az adott rendszer teljes fejlesztését egy külső cégre bízza. Ebben az esetben végképp a külső cég becsületességén és piaci megbízhatóságán múlik, mikor mennyit számláz ki – a kód itt is zárt, és a felület még azt a biztonságot sem adja meg, hogy „a dobozos terméket legalább többen használják, így ha valami gond van vele, a gyártó jó eséllyel már rendelkezik megoldással. Sőt ha az elején elfogadott rendszertervből hiányzik mondjuk a gumipitypang számára fontos hőmérséklet figyelése, könnyen kaphat a cég a beszállítótól pármillió forintos ajánla-

tot egy hőmérsékletet-is-figyelő programváltozat elkészítésére. Ennek pedig többnyire az a vége, hogy a cég felvesz egy nyugdíjast, akinek az a feladata, hogy óránként megnézze a hőmérőt. Lényegében a harmadik változat tűnik a legérdekesebbnek, feltéve, hogy a vállalat az Ég vezérletével megtalálja azt a pár programozót, akik pont az ő számára megfelelő eszközökkel szeretnek dolgozni, rendszerszervező ismeretekkel

is rendelkeznek, remek kódolók és hosszú távon egy helyen akarnak dolgozni. Mondjuk fél év alatt kifejlesztik a Pitypang 1.0 változatot, ami már működik is, és folyamatos üzemeltetés mellett igény szerint továbbfejlesztik. A veszélyeket itt is nyugodtan sorolhatjuk. Egyrészt a megfelelő ember megtalálása nem két nap. Másrészt szakember kell ahhoz, hogy eldöntsük, valóban jó embert találtunk-e (és a szakember szintén tévedhet); harmadrészt, ha tévedünk, azt csak fél, egy, vagy éppen két év múlva tudjuk meg. Negyedrészt fent kell tartanunk egy teljes fejlesztőkörnyezetet, és sajnos még fehér hollónak számít az a programozó, aki egy ilyen Pitypang 1.0 rendszert esetleg GNU-eszközökkel akar fejleszteni.
Tehát mint láttuk, mindhárom modellnek komoly veszélyei vannak. Nem beszélve arról, hogy mindhárom modellnél szerepel még egy erős veszélyforrás: mi van, ha a beszállító csődbe megy, és úgy dönt, hogy nem nyújt megfelelő fejlesztést, támogatást vagy a programozó feláll az asztaltól, mert jobb ajánlatot kapott egy Borókaföld 1.0 megírására? Baj. Méghozzá nem kicsi. Az amerikaiak jellemző módon erre a helyzetre is kitaláltak egy nevet, úgy hívják: kamionhatás: „Mi lesz, ha a programozónkat, aki az egész rendszert fejlesztette és üzemeltette, holnap elcsapja egy kamion?”
A három modell tulajdonságait és veszélyeit szem előtt tartva több cég egy negyedik modellt, a nyílt fejlesztésű modellt választotta, melynek kialakulásához tehát piaci igények vezettek.

Melyek ezek elsősorban? Először is a vásárló biztos akar lenni, hogy nincs kiszolgáltatva egy másik cégnek vagy személynek, aki jogokkal, kiváltságokkal rendelkezik a cég által használt (esetleg üzletileg érzékeny) program felett. Erre az a legjobb megoldás, ha a program kódja a cég tulajdonában van, esetleg olyan kód, amire biztos, hogy „örökéletű” fejhasználási joga van. Ez a legfontosabb pont.
Másodszor igényli, hogy a számára fontos feladatokat elvégezhesse, illetve igény szerint továbbfejleszthesse, kiegészíthesse. Mégpedig úgy, hogy ezért ne kelljen külön milliókat fizetnie, fejlesztőkörnyezetet vásárolnia, szakembert bérelnie vagy kitanítania; harmadszor pedig a rendszer felügyelete házon belül megoldható legyen, igény esetén pedig külső segítség is elérhető legyen.
Jön tehát a tanácsadó, végighallgatja szegény igazgatót, majd együttérző vállveregetés mellett a következő ajánlattal áll elő: „Figyelj, Géza! Keresünk még két-három olyan céget, akik hasonló cipőben járnak, mondjuk egy dísznövényest, meg valami tökös gazdát, összehozlak titeket, és elkészítünk egy olyan rendszert, ami mindhármótok igényét kielégíti. Így igaz, hogy három helyzetfelmérés van, de már csak egy rendszertervet kell készíteni, nem is beszélve arról, hogy a kipróbálás párhuzamosan mehet, így a kényesebb feladatoknak csak az egyharmada terhelődik rád. Sőt az összköltségnek is csak az egyharmadát kell állnod. A teljes fejlesztést GNU-eszközökkel végezzük, és utána a rendszert kódostul, kutyafüléstül mindhárman megkapjátok.”
Ez is egy lehetőség, de most jön az ötödik változat, a tényleges „nyílt rendszer”: „Mondok neked még egy lehetőséget. A fejlesztés után az alapkódot nyissátok meg a külvilág felé! Így amit elkészítettünk, azt a világon bárhol használhatják, ahol a tiedhez hasonló feladatokat kell ellátni. Hogy ez miért jó neked, hiszen minden ellenfeled is elkezdheti használni? Hát éppen azért jó! Na figyelj, ha Kelet-Üzbegisztánban is ezt a rendszert kezdi el használni egy pitypangos, és neki szüksége van egy páratartalom-vezérlő modulra is, megíratja magának, majd felküldi ezt is a rendszerbe. Te nem is gondoltál rá, hogy a páratartalomra is figyelj, de ott a modul, kipróbálod, és rájössz, mennyivel szebbek lesznek a pitypangjaid. Kész nyeremény, nem?”
Természetesen ez az elmélet sok kérdést felvet; a támogatást másként kell megoldani, a követés is más rendszert igényel, valamint a hivatalos szervek felé is másképp kell fordulni. Nincs sajnos hírem olyan számlázórendszerről, amelyet az APEH elfogadott volna, és a forrása nyílt volna. De kíváncsi vagyok, vannak-e olyan cégek idehaza, akik nyílt fejlesztésben gondolkodnak? Mondjuk egy nyílt számlázórendszeren?

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

Felhívás!
Ha vállalatod nyílt fejlesztésen gondolkodik, mondjuk egy számlázó- vagy raktárkezelő rendszer kifejlesztésén, kérlek, írj nekem! Hátha össze tudjuk hozni! Szívesen adnék hírt az első ilyen típusú komolyabb fejlesztésről!

A leszármazott alkotásokról

Mikor mondhatjuk, hogy egy program egy másikból származik?

A nyílt kódú programok sok felhasználóját megrémíti a „leszármazott alkotás” kifejezés. Aggódnak, hogy véletlenül az eredeti programból származó változatot hoznak létre, és így saját jogdíjas programjukat egy nyílt forráskódú program terjesztési engedélyének hatálya alá helyezik. Ez összetett kérdés, amellyel kapcsolatban a bíróságok és jogászok álláspontja nem egységes, de úgy gondolom, kereshetünk olyan meghatározásokat, amelyek segítenek a kételyek eloszlatásában.

Először is egy rövid emlékeztető arról, miért olyan fontos kifejezés a leszármazott alkotás. A terjesztési engedélyekben jellemzően valami ehhez hasonló áll: „A terjesztő ezúton engedélyezi … az eredeti program alapján készült, abból leszármazott alkotások elkészítését és terjesztését, ***feltéve, hogy ezen leszármazott alkotás példányait jelen engedély feltételei mellett terjesztik***.” Lásd például a GNU General Public License-t (➲ http://www.gnu.org/licenses/gpl.html). Miként állapítható meg, hogy leszármazott alkotást készítettünk-e? A szerzői jogi törvény szövege meglehetősen általános, és a programkódról egyáltalán nem esik szó benne:

> **A „leszármazott alkotás” egy vagy több létező mű alapján készült alkotás, például fordítás, zenemű, színpadi változat, irodalmi adaptáció, egyszerűsített változat, filmváltozat, hangfelvétel, képzőművészeti ábrázolás, rövidítés vagy bármely más forma, amelyben egy mű visszaadható, átalakítható vagy adaptálható. A szerkesztői javításokat, jegyzeteket, magyarázatokat vagy más, szerzői alkotásnak minősülő változtatásokat tartalmazó mű is „leszármazott alkotás”.** *(17. U.S.C. § 101)*

Szinte mindenki egyetért abban, hogy egy szerzői jogi védelem alatt álló program kódjának fizikai megváltoztatásával – ha a programot átírjuk vagy más gépi nyelvre fordítjuk – az eredeti program leszármazott változatát hozzuk létre. Ez az egyszerű eset. Ha ezt egy GPL vagy OSL engedéllyel terjesztett programmal tesszük, tiszteletben kell tartanunk a terjesztő feltételeit, és nyilvánosságra kell hoznunk az általunk terjesztett programváltozat kódját.

Mi történik azonban akkor, ha az eredeti programot pusztán lemásoljuk, és azt a saját, esetleg nagyobb méretű programunk összetevőjeként használjuk föl? Lényeges-e az, hogy nem változtattunk a kódon ahhoz, hogy az eredeti programot saját, nagyobb programunkba illesszük? Ha egy programot az eredeti programmal csupán összefűzünk, anélkül téve ezt, hogy megváltoztatnánk annak kódját, leszármazott programot hozunk-e létre? Szinte minden program használ programkönyvtárakat. Az összes programozó nyilvánvalóan nem készít a könyvtárból leszármazott művet azáltal, hogy meghív belőle egy `sqrt` függvényt. Miért lenne más a helyzet, ha olyan összetett programokat hívunk meg, mint amilyen egy vállalati döntéstámogató rendszer vagy egy adatbázismotor? Mi a helyzet akkor, ha egy GPL vagy OSL engedéllyel terjesztett programba szerkesztjük bele a saját programunkat, például egy eszközmeghajtót a Linux-rendszermagba?

Számít-e, hogy az összeszerkesztés mely technikáját alkalmazzuk? Vajon egy ilyen elemzés (statikus, dinamikus összeszerkesztés, adatátadás API-n keresztül stb.) szakmai jellegű szőrszálhasogatás, ami mindössze arra jó, hogy szem elől tévesszük a lényegi kérdést? Hogyan tarthatna lépést a leszármazott alkotásokról szóló joganyag a programkészítés technológiai fejlődésével? Ezek a kérdések azért fontosak, mert egyes terjesztési engedélyek megkövetelik, hogy a létrejött leszármazott programban lévő saját kódrészeinket is nyilvánosságra hozzuk. Ezt a kötelezettséget nem minden esetben vagyunk hajlandók vállalni. A fent leírt esetekben a következők alapján döntenék.

1., A LEGFONTOSABB jellemző annak megítélésekor, hogy egy program leszármazott alkotás-e, az, hogy az eredeti program kódját az új program elkészítéséhez felhasználták, módosították, lefordították vagy bármilyen más módon megváltoztatták-e. Amennyiben nem, akkor megítélésem szerint nem leszármazott alkotásról van szó.

2.) A LESZÁRMAZOTT program kifejezés értelme nem tágítható ki olyan módon, hogy magába foglalja azokat a programokat, amelyeket erre a célra tervezett és alkotott programkönyvtárakkal szerkesztettek össze. Amikor például egy cég kiad egy természettudományos függvénykönyvtárat vagy egy objektumkönyvtárat, akkor azok, akik mindössze – a kódot esetleg meg sem tekintve – változtatás nélkül használják a könyvtárat, nem hoznak létre a könyvtárból származó alkotást.

3.) NEM TEKINTHETŐK leszármazott programoknak a bővítmények és eszközmeghajtók, amelyeket úgy terveztek, hogy készen kapott, változtatás nélkül használt programokkal szerkesszék össze őket. Ha egy GPL engedéllyel terjesztett programot úgy terveztek, hogy különálló bővítményeket fogadjon, nem hozunk létre leszármazott alkotást pusztán azáltal, hogy efféle bővítményt futtatunk a program felügyelete alatt, még akkor sem, ha meg kellett tekintenünk a kódot, hogy megtudjuk, hogyan illeszthető hozzá a programunk.

4.) A LEGTÖBB esetben nem kell a programok összeszerkesztésének szakmai részleteivel foglalkoznunk, kivéve, ha ez segít meghatározni, vajon a program készítői osztottak-e valamilyen fölismerhető közös elképzelést arról, hogyan is nézne ki egy leszármazott alkotás. Figyelembe kell vennünk az értékesítési módok apró különbségeit, amelyek a szerző szándékát jelezhetik. Ilyen például, ha a létrejött programot az eredeti „bővített” változataként árulják, vagy amikor az eredetit úgy tervezték és reklámozták, mint ami „programkönyvtárként” fejleszthető.

Linux Journal 2003. január, 105. szám

Lawrence Rosen
(www.rosenlav.com) Magángyakorlatot folytató jogász. A Nyílt Forrás Kezdeményezés igazgatatója és jogtanácsosa.
➲ http://www.opensource.org

A Microsoft Exchange működése és helyettesítése

A talán legnehezebben helyettesíthető Microsoft-kiszolgáló a drága és kiábrándító Exchange. Lássuk, mit kezdett vele az IBM és a Bynari.

A 2000. év vége felé az IBM bemutatta, hogy képes linuxos gépek ezreit futtatni egyetlen S/390-es nagyszámítógépen.
Ugyanakkor az IBM ráébredt, hogy a GNU-alkalmazások nem fedik le a teljes termékkínálatukat. Az IBM-nek olyan alkalmazásra volt szüksége, amely a Linuxot a számítástechnika főáramába jobban illeszkedővé emeli – ezt megvalósítandó megkerestek több partnert, többek között minket is.
Amikor 2001 áprilisában az IBM zSeries (a S/390 új neve) termék felkeresett bennünket a Bynarinál, nem igazán értettem az érdeklődését. A látogatás után azonban már tökéletesen tisztában voltam vele: a Bynari Inc. lesz az IBM első linuxos tanácsadó partnere.

Az elektronikus levél, a „gyilkos alkalmazás"

Az emberek eredetileg úgy ismertek minket, mint akik a Microsoft Exchange kiszolgálóval beszélgetni képes ügyfeleket készítenek. Piacunk szélesítésének lehetőségeit keresve ráakadtam ***Johnson*** és ***Mead*** „Exchange Replacement HOWTO" című írására (➲ http://www.bynari.net/whitepapers_howto.html). Munkájukat felhasználva Linux-ügyfeleinknek és a Microsoft Outlooknak felépítettünk egy kiszolgálót. Kiszolgálónkat és a hozzá tartozó Outlook Configuration Guide-ot azután felkapták az értékesítői csatornák.
Amikor a kiszolgálókódot átvittük S/390 Multiprise 3000 alatt futó Linuxra, még senkit sem ismertünk az IBM-től. ***Jimmy Lee*** (akkoriban még az Equant színeiben) biztosította a forrásokat, hogy lássa, meg tudjuk-e csinálni. ***Gary Ernst*** az Equanttól beállította a Linux S/390-es változatát, és segített életet lehelni a kiszolgálónkba.
Volt tehát egy termékünk, amely lehetővé tette, hogy unixos és Outlook-ügyfelek találkozókat időzítsenek, és naptárfeladatokat továbbítsanak, amennyiben a Microsoft Outlook Mail Only módban maradt, valamint egyenrangú hálózati (peer-to-peer) könyvtármegosztást alkalmazott. Kiszolgálónk szépen működött, szimulálni tudtuk az Exchange teljes körű címlistáját (Global Address List, azaz GAL), miközben meg tudtuk jeleníteni a felhasználók szabad, illetve lefoglalt idejét, és egész tűrhető karbantartói felületünk volt.
A Microsoft azonban piacra dobta az XP-t, és komolyabb változtatások történtek az Outlookban. Termékünknek hirtelen kiszolgálóoldali naptárkezelésre lett szüksége. Attól tartottunk, hogy az IBM vállalkozókból álló vásárlóinak az Outlook alacsony költségű kiszolgálómegoldásai iránt érzett növekvő kedve esetleg semmivé apadhat. Ezért szükségünk volt az Outlook Internet Mail Only módja alatti egyenrangú hálózati naptármegosztásra, vagy kezdenünk kellett valamit a kiszolgálóoldallal.

A megoldás megszületésének 45 napja

A 2001 januárjában tartott LinuxWorld Conference & Expo eljövetelekor az IBM az Outlook XP-felhasználókkal mit sem törődve folytatta kiszolgálónk árusítását. Tudtam, hogy valamit tennünk kell, mégpedig gyorsan, mert különben elveszítjük az IBM bizalmát. Ez idő tájt a piaci gondot leginkább átlátó személy ***Roger Luca*** volt a Mainline Information Systemstől. Szerencsére Roger és én régi jó kapcsolatban álltunk egymással. Rogerrel a marketing és az eladási részleg élén a Mainline az IBM szuperszámítógépek legnagyobb viszonteladója lett, egyben ők voltak az IBM utáni legnagyobb támogatóink. Roger ellátott bennünket alkatrészekkel, hogy termékünkbe beépíthessük a kiszolgálóoldali naptárazást, illetve embereket biztosított, ha alkatrészgondokkal kerültünk szembe.

A fejlesztőcsapat megdöbbenése

Képzeljük el, hogy éppen most fejeztünk be egy egyéves kimerítő fejlesztést. Már le is foglaltuk az utakat és egyéb terveink vannak a szabadságunkra, amikor hirtelen csörög a mobil: főnökünk fontos találkozóra hív bennünket. Nos, éppen ez történt a boltunkban. Szinte hallottam a döbbenetet a fejlesztők hangjában, ahogy válaszoltak a hívásomra.
2001. november 7-én találkoztunk, hogy lássuk, ki tudunk-e adni karácsonyig egy kiszolgálóoldali naptármegoldást.
A Mainline-nak volt néhány függőben maradt eladása, és szükségük volt erre a szolgáltatásra.

A szakmai kihívások nem az ijedőseknek valók

Vezető fejlesztőként én készítettem el a munka keretrendszerét. Elméletben a feladat különféle szakaszokra és feladatokra bontható. Hogy a projekt életét megkönnyítsük, háromlépéses megközelítést vezettem be, ezeket kutatásnak, befektetésnek és végrehajtásnak neveztem el. Minden szakaszhoz egy-egy mérföldkövet rendeltem. Így aztán a kód helyett elsőként az Internet és a fellelhető könyvek böngészésével nyitottunk.
Egy hét kiadós kutatás után feltérképeztük a nehézségeket. Ki kellett találnunk, hogy a Microsoft DCE-RFC protokollja hogyan tárolja és mozgatja a naptárbejegyzéseket. Értelmeznünk kellett a mozgatott adatokat, és olyan formátumba kellett szervezni, amelyet IMAP-kiszolgálón tudunk tárolni, ahonnan az adatot a megszokott módszerrel az Outlook-ügyfélhez lehet továbbítani. Az adatsémákon túl eléréskorlátozást is be kellett vezetnünk, hogy a felhasználó a saját naptárában különböző engedélyszinteket adhasson meg és oszthasson ki a többi felhasználónak.
Egy újabb kemény munkával töltött hetet áldoztunk a kutatásra, végül megállapodásra jutottunk. Minden szakértő, hírcsoport, Outlook-szakértő és cég, aki csak megpróbálkozott a feladattal, azt állította, hogy képtelenség Microsoft Outlook naptárszolgáltatást készíteni IMAPkiszolgáló alatt. Nos, mi eláruljuk, hogyan is sikerült.

Felejtsük el az átvitelt, és összpontosítsunk az adatra!

Először is felfedeztük, hogy a Microsoft Exchange-modell munkájának 95 százalékát az Outlook végzi. Az Exchange (magyarul csere) – mint a neve is mutatja – az Outlook-felhasználók közti adatcserét bonyolítja le. Ennek megfelelően az Exchange kezelte az átviteli protokollt is, és ez okozta a gondot. Az emberek általában a MAPI rövidítést használják, ha az Outlook-fel-

használók közti adatcserét lebonyolító Exchange-protokollról esik szó. Nos, én nem találtam ilyen értelemben MAPI-t. Amikor elfogtuk az Outlook-üzeneteket, szöveg helyett jókora, bináris adatot tartalmazó tömböket találtunk. Felismertem az adatot, de nem rémlett, hol is láttam azelőtt.

Egyetlen szuperszámítógép Linux-példányok ezreit képes futtatni

Az Outlook különböző arcai

Az Outlook két különböző módban futhat: ***Corporate Workgroup*** (üzleti munkacsoport) és ***Internet Mail Only*** (csak internetes levelezés) módban. Corporate Workgroup módban a Microsoft bekapcsolja az Outlook nagyra értékelt képességeit. Az Internet Mail Only módban a Microsoft egy teljesen másik, minden leírást nélkülöző alkalmazási programfelületet (API) használ, korlátozott képességkészlettel. Exchange kiszolgáló nélkül az Outlook egyáltalán nem működik Corporate Workgroup módban, és csak csökkentett képességekkel bír.

Az Outlook Workgroup módjában, vagy amikor az Exchange kiszolgálóhoz csatlakozik, az adatok bináris formátumban cserélődnek. Ezt a bináris adatot egységesített figyelővel lehetetlen felismerni.

A kutatás során a Sourceforge.net-en találtam egy fejlesztőhelyet, ahol az Open DCE-t alakították át Linux alá. Írtam egy levelet az egyik fejlesztőnek, aki elmondta, hogy a kódot az Open Group adományozta.

Elmentem az Open Group weblapjára, és végignéztem az anyagokat, találtam is egy régi cikket, amelyben megemlítik, hogy a Microsoft engedélyeztette a DCE-t. Letöltöttük az Open DCE-kódot, majd a motort felhasználva összekapcsolódtunk az Outlookkal, majd az Exchange-dzsel. Végre többet tudtunk az átviteli protokollról, ugyanakkor a bináris adatfolyamok jelenlétét is megértettük.

Felfedeztünk tehát, hogy a Microsoft az osztott számítási környezetet (Distributed Computing Environment, azaz DCE) használja fel a szállításra, amikor az Exchange-et és az Outlookot Corporate Workgroup módban használjuk. A Microsoft programozási felületet ad a DCE-hez, és ezt nevezi MAPI-nak. A MAPI alatt azonban továbbra is egy nyílt szabványokon alapuló protokoll (DCE) bújik meg, amelyet a Microsoft az Open Grouptól vásárolt meg, majd módosított.

A DCE egyik alapértelmezett függvénye a szövegeket önműködően bináris objektumokká alakítja. A Microsoft ezt a bináris objektumot dokumentálatlanul hagyta. Így aztán a MAPI-tulajdonságok programozói részeinek többsége általuk sem felismerhető bináris kódként végezte. Hogy a dolgokat egy kicsit még jobban összezavarják, a Microsoft a bináris tulajdonságkódokat jókora bináris nulladatba ágyazta, így próbálván elrejteni azt.

Már kezdtük megérteni az átvitelt, amikor rájöttünk, hogy az Outlook MIME-csatolmányokat küld a többi Outlook-ügyfélnek. Ezek a csatolmányok nem alakították át magukat bináris adattá. Arra a következtetésre jutottunk, hogy az Outlook a csatolmányok átvitelére egyfajta beágyazást használ – ez vezetett el bennünket a TNEF objektumig.

TNEF

A Microsoft Exchange számos programot használ, ezeket ***service provider***-nek (szolgáltatásszolgáltatónak) nevezi, a Linux-felhasználók pedig démonnak hívnák. A külön viselkedéssel és állapottal rendelkező objektumokat az Exchange szolgáltatásnyújtói (service provider) kezelik.

A Transport neutral encapsulation format (TNEF, magyarul szállítássemleges beágyazási formátum) olyan módszer, amellyel ASCII-szöveget, más fájlokat és objektumokat a bináris üzenetadattal együtt lehet átadni. A bináris üzenetadat képezi a TNEF objektum érdemi részét. A TNEF bináris folyamba ágyazza be a MAPI-tulajdonságokat, amely a szállítás és az átjárók során az üzenettel együtt utazik. Az Outlook képes visszakódolni a beágyazást és értelmezni az eredeti üzenet összes tulajdonságát. A TNEF objektum csatolmányként rejtőzik MIME alatt.

Miután megtaláltuk a naptáreseményeket készítő tulajdonságokat, készítettünk egy TNEF-kódolót, és hamarosan már képesek voltunk SMTP-n keresztül naptárüzeneteket küldeni az Outlook-ügyfeleknek és -ügyfelekre. Azonnal felismertük, hogy felhasználhatjuk az internetes szállítási protokollokat, és a Microsoft Corporate Workgroup módját a MAPI nélkül is bekapcsolhatjuk. Amikor aztán megláttuk a Microsoft Knowledge Base Article Q197204 cikkét, amelyben kijelentik, hogy a Microsoft átviteli protokollunkat Workgroup módban nem támogatja, már tudtuk, hogy célnál vagyunk.

Exchange-ügyfélbővítés

Minthogy elsődleges célunk a kiszolgálóoldali naptárkezelés volt, készítenünk kellett egy üzenettárat, ahol az Outlook-ügyfelek objektumait tárolhattuk. Mivel IMAP-kiszolgálót használtunk, IMAP-támogatásra volt szükségünk, amit azonban a Microsoft Workgroup módban nem támogat. Ezért meg kellett találnunk a módját, hogyan ruházhatjuk fel az Outlookot IMAP-támogatással.

Amikor a Microsoft első ízben adta ki az Exchange-t, az Outlook még nem is létezett. Helyette a Microsoft Exchange üzenetküldő ügyfelek készletét nyújtotta különböző Windows operációs rendszereihez. A Microsoft ezeket az Exchange üzenetküldő ügyfeleket bővíthető szerkezettel látta el. Az ügyfélbővítés lehetővé tette a fejlesztők számára, hogy megváltoztassák az Exchange ügyfél alapértelmezett viselkedését. Amikor a Microsoft kiadta az Outlookot, megtartotta az Exchange-ügyfélbővítések támogatását, hogy a már létező ügyfélbővítési DLL-ekkel továbbra is együttműködhessen.

Az ügyfélbővítés lehetővé teszi, hogy megváltoztassuk az ügyfél alapértelmezett viselkedését. A Microsoft előnyösnek találta a kiterjesztéseket, mivel közvetlenül az ügyfélhez kényelmesen lehetett új testreszabott szolgáltatásokat adni vagy a viselkedését megváltoztatni, s nem kellett külön MAPI-alkalmazást készíteni. A mi szemünkben viszont a kiterjesztések egyedülálló lehetőséget kínáltak, hogy az IMAP ügyfélszolgáltatást az Outlook Workgroup módjához adhassuk. Ezzel a szerkezettel parancsokat adtunk az Outlook menühöz, saját gombokat helyeztünk az eszközsorba, és az IMAP ügyfélszolgáltatások segítségével lehetővé tettük a kimenő és bejövő levelek előfeldolgozását.

Szerencsére már korábban elkészítettük az IMAP-ügyfélkönyvtárakat, amikor tavaly a linuxos ügyfelünket készítettük. Most egyszerűen csak át kellett írni Windows alá. A függvényhívások, fejlécek, és protokollok ismerete erőfeszítéseinket a felére csökkentette.

Miután elkészítettük az ügyfélszolgáltatásokhoz tartozó Microsoft DLL-t, Outlook-kiterjesztésként beillesztettük.

Szerencsére már az első alkalommal működött. A levelekhez és meghívókhoz a rich text formátumot (RTF) választva, TNEF objektumunk csatolta magát a levélhez. Mivel a TNEF objektumot az Outlook készítette, gond nélkül küldözgette őket. Ezt követően üzeneteinket feltöltöttük IMAP könyvtárunkba a Microsoft ***.pst*** fájlját használva tárolásra és csereterületként. Mivel kapcsolatban maradtunk az Exchange-dzsel, és a kiszolgálónkat tudtuk üzenettárolásra használni, láttuk, hogy a két rendszer együttműködik egymással. Exchange könyvtárunkból átrántottunk néhány objektumot IMAP könyvtárunkba. Ahogy ezt megtettük, láthattuk, hogy a feladatok, naplóbejegyzések, naptáresemények és egyebek mind-mind felbukkantak az Outlookban, pontosan úgy, mintha Exchange-en keresztül érkeztek volna. A naptár úgyszintén hibátlanul működött.

Az Exchange

Ha megnézzük az Exchange-et és megszámoljuk az összetevőit, látni fogjuk, hogy mindössze négy darabból áll. Az első az adatraktár vagy üzenetraktár. A raktár tartalmazza az egyes felhasználók üzeneteit, és a hozzájuk rendelt elérhetőségi lista (Access Controll List, azaz ACL) motort. Az RFC-megfelelő IMAP-kiszolgálókhoz hasonlóan a névtér eltérő lehet, attól függően, hogy a raktár egyedi felhasználókhoz tartozik-e vagy nyilvános.
A raktár tárolásához a Microsoft Access adatbázist használ. A Microsoft Jet Engine technológia korlátai és az Access MDB fájlok megakadályozzák a vertikális méretezhetőséget.
Másodszor az Exchange címtárral rendelkezik. A Microsoft objektumosztályok és tulajdonságok alapján rendszerezte az Exchange címtárát. Az Exchange könyvtárszerkezet hasonlít az RFC-megfelelő LDAP protokollhoz. Csak éppen a Microsoft kibővítette az ***Object Class***-okkal, és ezekben, illetve a többi osztályban kicserélte a tulajdonságneveket.
Ezen kívül az Exchange-ben van egy levéltovábbító (Mail Transfer Agent vagy MTA). A Microsoft MTA hasonlónak tűnik a korábbi, Microsoft Mail 3.5-nek nevezett termékeiben használt MTA-hoz. A Microsoft Mail MTA-nak csatlakozókra vagy kapukra van szüksége, amelyek olyanra írják újra a saját szabadalmazott levélfejlécét, amilyen már együtt tud működni az idegen rendszerekével, ilyen például a Lotus Notes, az X-400 vagy az RFC 822 internetes levélszabvány. A Sendmaillel és a hasonló internetes MTA-kal ellentétben az Exchange MTA-jának nincsenek beállítási lehetőségei.
Végül az Exchange rendelkezik egy rendszerkezelőnek (system attendant) nevezett résszel. A kezelő foglalkozik az Exchange-ben végzett minden művelettel, a levélküldéstől és fogadástól kezdve a címeknek az Exchange-címtárból történő kitöltési kérelméig. A rendszerkezelő sok szempontból olyan, mint próbálkozás a folyamatközi kapcsolattartás (IPC) megvalósítására, ami a Microsoft operációs rendszereiből hiányzik.

A Microsoft túlszárnyalása Berkeley-adatbázissal

Linux kiszolgálóoldali megoldásunk az Exchange-ben alkalmazott elemekhez hasonlóan épült fel. Itt van mindjárt a Cyrus IMAP-üzenettár. A Cyrus-raktárak egyedi felhasználói üzeneteket tárolnak és hozzájuk rendelik az ACL-motort. A névtér aszerint különbözik, hogy a raktár egy adott felhasználóhoz tartozik-e vagy nyilvános. A Cyrus a Sleepycat Software, Berkeley adatbázisát használja. Ahol a Microsoft Jet Engine és Access adatbázis technológiája a méretezhetőség gátja volt, a Berkeley adatbázis magas teljesítménye és méretezhetősége folytán felhasználók ezreinek munkáját képes támogatni akár 256 terabájtos adatbázisokon is.
Másodszor, a Linuxnak van címtára. Míg a Microsoft az Exchange címtárát szerkesztette a Lightweight Directory Access Protocol (LDAP) szabványnak megfelelőre, a Linux-megoldás az OpenLDAP programot, az LDAP nyílt forrású változatát használja. Az Outlook-ügyfelek elhelyezéséhez az Exchange objektumosztályokat és nem együttműködő névváltozataikat be kellett illesztenünk. A Microsoft alapú azonosítókhoz bevezettünk egy indexet, és nagy sebességű átfogó címlistát hoztunk létre.
Akárcsak az Exchange, a Linux is rendelkezik MTA-val, amelyet viszont belsőleg lehet beállítani és kezelni, és nincs szüksége külső kapcsolódási pontokra. A Cambridge-i egyetem által fejlesztett Exim nevű linuxos MTA-t használtuk. Az Exim-hez számos beállítási lehetőség tartozik, többek között keresőfájlok, helyi kézbesítés, szabványoskifejezés-támogatás.

Az Exchange lecserélése

Johnson és Mead a kiszolgálóoldali üzenetváltást és a karbantartói konzol elkészítését a Linux-fejlesztők következő nemzedékére hagyta. Mi két lépésben hajtjuk végre a feladatot: először elfogjuk az Outlook-üzeneteket és visszafejtjük a TNEF objektumokat. Másodszor az Exchange-ben használt ügyfélkiterjesztési szerkezet segítségével IMAP-lehetőségeket biztosítunk a Corporate Workgroup módban futó Outlook számára.
E két lépés után a programozó vagy az időszakos rendszergazda választási lehetőséget kínáló szolgáltatásnyújtót (service provider) készíthet az Outlookhoz, és számos hagyományos levelezőügyfelet támogathat. A Linux-levélkiszolgálók nem maradnak el a más felületeken használtaktól. Használhatjuk a Netscape Mailt, az Outlook Expresst, a Ximian Evolutiont, vagy a Pine-t.
Az olyan magas méretezhetőségű Linux-összetevők, mint az Cyrus IMAP, az OpenLDAP és az Exim egyetlen Intel-gépen Exchange kiszolgálók tucatjait képesek helyettesíteni. A többrétegűség és a ma már túlhaladott DCE-összetevők nem akadályozzák a Linuxot.
Ha grafikus karbantartói konzolt keresünk, a PHP Cyrus eszközök, `cyrus_imap-sql`, Webmin és Replex rendszerek a felügyeleti munkát gyerekjátékká egyszerűsítik.
Általában véve kevés ember gondolná, hogy az Exchange linuxos kiváltása könnyű feladat. Ennek ellenére fejlesztőcsapatunk megmutatta, hogy igenis meg lehet tenni. Remélhetőleg sikerült eloszlatnunk az Exchange kiszolgáló körüli ködöt, és megmagyaráznunk a rémisztgető szóbeszédek nagy részét.

Linux Journal 2003. február, 106. szám

Tom Adelstein (tadelstein@xandros.com)
Texasban a Xandros Inc.-nek dolgozik, a cég dallasi kiszolgálórészlegének a feje. Jelenleg elsősorban a webszolgáltatások és a Xandros Linux asztali gép támogatásának a területe érdekli.

KAPCSOLÓDÓ CÍMEK

Bynari InsightServer for Linux x86 és zSeries gépeken
➲ http://www.bynari.net/insightserver.html
Cyrus IMAP ➲ http://asg.web.cmu.edu/cyrus
Exchange Server Replacement HOWTO
➲ http://www.arrayservices.com/projects/Exchange-HOWTO
Exim ➲ http://www.exim.org
OpenLDAP ➲ http://www.openldap.org

Linux alapú irodai hálózat megvalósítása (2. rész)

Az előző részben felépítettük a rendszert, munkába is állítottuk, most pedig ülünk a kedvenc gépeink között, telve a fáradtság, büszkeség és a kíváncsiság keverékével. Lássuk, milyen tapasztalatok gyűltek össze az azóta eltelt időben.

A rendszer a próbaüzemet nem számítva (tehát „élesben") immár több mint egy hónapja üzemel. Ennyi idő a legfontosabb tapasztalatok megszerzésére talán már elegendő, természetesen annak elfogadása mellett, hogy ezt a rendszert folyamatosan lehet építeni, fejleszteni, és épp ezt szeretjük benne annyira.

Az elmúlt hónap kitermelt magából egy felülvizsgált új rendszerlenyomatot, amelyben benne foglaltatnak olyan alapvető, de általában (illetve általam) elfelejtett programok, mint a grafikus tömörítő (Ark, melyhez hasonlatosat én még sose használtam Linux alatt); a Windows alatt elterjedt tömörített formátumok tömörítőprogramjai (zip, rar, arj); grafikus eszköz a CD-ROM és a hajlékonylemez befűzésére (kwdisk), vagy néhány, a használatot és a telepítést megkönnyítő közvetett hivatkozás. Mint említettem, én a saját gépemen elég régóta Linuxot használok. Amiben ez a kísérlet számomra merőben új, az az, hogy mindent grafikusan kell megoldani, hiszen a gépet átlagos felhasználók fogják használni. Amikor a saját rendszeremen belefutok valamilyen hibába, a megoldás szépségét pont az adja, hogy a javításokat konzolról lehet eszközölni. Itt viszont azt kell szem előtt tartani, hogyan lehet a billentyűzet kizárásával, csak ide-oda kattintgatva elérni azt a szolgáltatást, amit szeretnénk. (Ez némileg szemben áll az „egérhez nem nyúlunk" alapelvvel…)

Mindez azt eredményezi, hogy bármennyire is húzódozom az ilyesmitől (szerencsére azért annyira nem), mivel a felhasználók fogják használni, kénytelen vagyok kérdezni, illetve odafigyelni rájuk. (Ez nehezebb, mint gondoltam, bár nagyon kedves lányok is dolgoznak a cégnél…) Nem dédelgettem ábrándképeket, hogy csupa hasznos tanáccsal fognak ellátni. Természetesen sokan mindig is hibaként fogják elkönyvelni, ha nem találnak egy szolgáltatást az OpenOffice.org Writerben, ami egyébként ott van, csak két menüponttal arrébb, mint megszokták. De ha az ötven hibajelentésből csak egyetlen tartalmaz is építő jellegű tanácsot, már előrébb vagyunk (a saját esetemben szerencsére ennél több akadt).

A felhasználók panaszai legtöbbször a sebességre vonatkoznak. Ez egyrészt szomorú, mert igaz: egyes alkalmazásokkal valóban gondok vannak ezen a téren, másrészt tudni kell, hogy Windows alatt is ez volt a legnagyobb baj. Saját próbáim alapján a rendszer sebességével általában véve ezeken a vasakon nincsen gond. Az OpenOffice.org azonban sajnos még a gyorsítóval sem éri el a Microsoft Office sebességét, és mivel a felhasználók még nem fedezték fel az előbbi előnyeit (például a kezelhető dokumentumok széles kínálatát, vagy a külön szolgáltatásokat, mint például a valóban csodálatos Tündéreket), ebből lett a legnagyobb mumus. Ma az egyetlen nehézség rászoktatni a felhasználót arra, hogy ne csukja be az OpenOffice-t. Ez azzal is jár, hogy el kell neki magyarázni a „nem fog lefagyni" fogalmát, ami számára eddig ismeretlen volt. Nehezebb mint gondolnánk. A rendszer bemutatásakor külön fel kellett hívnom a figyelmet arra, hogy a linuxos gépet nem kell kikapcsolni („nem arra készült"), csak ki kell jelentkezni belőle. Mindamellett a Linux „hibája" az, hogy nem kelti azt az érzetet, mintha gyorsan dolgozna, nem adja a „sebesség illúzióját". Ha az ember elindít egy alkalmazást Windows alatt, első lépésként kap egy homokórát, aztán nyitóképernyőt, aztán a következő nyitóképernyőt, és így tovább, és nem hiszi azt, hogy „a gép elfelejtette, hogy elindítottam valamit". Ezzel szemben Linux alatt az alkalmazás nem dob ki fölösleges processzoridőt a csillogó-villogó, „most már aztán tényleg mindjárt" ablakokra, hanem csak elindul, előbb-utóbb. Ettől aztán felhasználó megijed, és kattint még négyet, amitől szép lassan betelik a memória, „elfogy" a processzoridő, és az ember kézzel lövögetheti ki a fölösleges folyamatokat.

A másik mumus meglepetésemre az Evolution levelező lett – én ugyanis már több hónapja (a legelső próbák ideje óta) használom gond nélkül a programot. (A használat itt többnyire napi 16–20 óra futást, és legalább 8 óra tevékeny üzemeltetést, tehát levélírást, naptárszerkesztést stb. jelent). Az látható, hogy az Evolution az IMAP terén nem áll maradéktalanul a helyzet magaslatán. Mint már említettem, ez nagyrészt betudható a meglehetősen nagy (pár száz megabájtos) IMAP-könyvtáraknak; ugyanakkor nem magyarázza, hogy Outlook Express alatt miért nem kap a kedves levelezőügyfél félóránként hibaüzenetet. (Természetesen a kép itt is árnyaltabb: Outlook alatt is vannak hibák, csak az nem szól róla – egyszerűen nem működik.) Az Evolution nagy előnye viszont az Outlookkal szemben az, hogy nem kell kézzel frissíteni az IMAP-könyvtárak listáját, mert induláskor ezt magától megteszi.

A megoldást itt nem az Evolution, hanem új IMAP és POP3 kiszolgáló telepítése adta. A korábbi UW-IMAP-ról Courier-re való frissítés a jelek szerint megszüntette a cikk előző részében említett „Broken Pipe"-okat.

További apró, ámde gyakran zavaró gond az Internet Explorerre egyszerűsített, rosszul megírt weboldalak meg nem jelenése a linuxos böngészőben. Ez nagyon fájó pont, mivel nem a mi böngészőnk a hibás, és nem tudunk mást tenni a kiküszöbölésére, mint erős „negatív" hullámokkal bombázni az illető oldal készítőjét. Ugyanakkor a valóságban kell élnünk: a helyzet adott, a főnököt pedig nem érdekli, miért nem látszik az oldal; ő csak azt látja, hogy ami működött Explorer alatt, az itt nem megy. Ilyenkor aztán az ember (a rendszergazda) magyarázhat szabványokról, meg böngészőháborúról; az ilyen esetek sajnos mind rosszpontot jelentenek a Linuxnak. Ez sajnos nem az egyetlen Windows–Linux ellentétből származó galiba: a rosszul megírt makróktól, egyes Office-ban mentett fájloktól elhalálozó OpenOffice.org vagy QTCups is ezt a vonalat erősíti. Remélhetőleg az idő előrehaladtával és a Linux átfogó erősödésével, valamint irodákon belüli terjedésével helyileg egyre kevesebb ilyen kínos helyzet fog előállni.

Még egy, a Linux alapjaiból fakadó kényes pont: Linux alatt nem

minden billentyűparancs működik ugyanúgy, mint Windows alatt, sőt bizonyos esetekben a QT és a GTK alapú programok kombinációi is különbözőek. Ez nekem nem gond, megtanulom mindet (ráadásul a saját gépemen nincs is QT); a felhasználónak viszont megint csak hiába magyarázom a két eszközkészletet és a köztük lévő különbséget: ő (rossz esetben) csak a káoszt szűri le belőle. Ez többnyire elkerülhető egy jó kézikönyv, illetve annak rövid kivonata segítségével.

Itt egy gondolat erejéig hadd térjek ki a használati útmutató kérdéskörére. Nagyon fontos a megfelelő leírás elkészítése. Tudom (hiszen én is így vagyok vele): dokumentációt írni unalmas feladat, viszont, mivel itt felhasználókról van szó, célszerű, ugyanis megállás nélkül kérdeznek. Sokszor teljesen közismert dolgokat, ráadásul nagyon sokszor ugyanazt. És nemcsak én unom meg a sok kérdést hamar (az első után), hanem ők is: mivel nem tudnak különbséget tenni a valódi hiba és kiküszöbölhető felhasználói balgaság között, a rendszer rossz színben fog feltűnni a szemükben. Más kérdés azonban, hogy elolvassa-e a felhasználó a dokumentációt, de ez már nem szakmai kérdés. Talán gonosz dolog ilyet mondani, de ha ott a kézikönyv, és még akkor is olyat kérdez, amire megvan a válasz, akkor kíméletesen el lehet magyarázni az „RTFM” jelentését, és senki nem fog minket hibáztatni…

Egy további apró nehézség a helyfoglalás: egy felhasználói könyvtár (user home directory) legkevesebb 27 MB a KDE és a többi csoda beállítóállománya miatt. Ez körülbelül ötven felhasználó esetében már nem kis helyet foglal a kiszolgálón, de tulajdonképpen annyira nem is sok, szerintem ennyit lehet áldozni a teljes testreszabhatóság oltárán.

A fentieken kívül a leggyakoribb „hiba” bizonyos programok hiánya volt, mint például az Arké vagy az Acrobat Readeré, ezek természetesen nem valódi hibák, csak az én feledékenységem eredményei, ezért hamar kiküszöbölhetők.

A kedvező vonásokat nem is annyira az alkalmazottak (ők még többnyire vizes kendővel törölgetik a homlokukat az átállás okozta sokk miatt), mint inkább én, illetve a felhasználótámogatást munkatárs érzi. Ezektől a kellemetlenségektől eltekintve ugyanis a rendszer üzembiztos, nincs összeomlás, illetve ha van (mert volt már az áttérés óta), az a vasra vezethető vissza. Ha a kedves alkalmazott nehézségbe ütközik, azt igen jó esélylyel távolról meg tudom oldani, és az általánosan előforduló hibák száma tulajdonképpen csekély: pár hetes használat után a gyakran előforduló kellemetlenségek elkezdtek ismétlődni, és nem nagyon jönnek elő újak.

A felhasználók kedvező újdonságként eddig többnyire csak azt vették észre, hogy az új rendszer általuk teljesen testreszabható, sokkal szebb, mint a régi, és személyre szól, vagyis nem „közös ló”. A lelkesebbek (őket elneveztük „PowerUser”-nek, illetve már egy „PowerUser+”-unk is akadt) nagy örömmel vetik bele magukat a háttér megváltoztatgatásába, fedezik fel például az átlátszó menü lehetőségét. Ez hatalmas haladás ahhoz képest, amikor hónapokkal ezelőtt, a váltás szóba kerülésekor rémülten kérdeztek vissza, mondván: „akkor majd mindent billentyűzetről, fekete képernyő előtt kell megcsinálni?”. A szívemet melengeti továbbá, ha hallom, hogy a felhasználók érdeklődnek a „társadalmi” háttér iránt. Most, felfedezve, hogy a Windowson kívül is van élet, hogy létezik választási lehetőség, páran olyan kérdésekkel jöttek hozzám, mint hogy miként is működik ez a szabad programmodell, milyen módon keletkezett a Linux. Mivel én valamennyire küldetésemnek érzem, hogy megszerettessem velük a Linuxot, egy ebéd keretében szívesen tartok mesedélutánt RMS-ről, Linusról, az eszméről, és látom, hogy lassan olvadozik a jég, és kezdik elhinni, hogy nincs az ügyben „átverés”, ez tényleg ingyen van, emberek tényleg lelkesedésből foglalkoznak vele, egy olyan rendszert építenek, ami a „mindenható” Windows babérjaira tör.

A jövő pedig hatalmas igéreteket tartogat, illetve lehetőséget egy kényelmesebb életre: itt nem áll fenn annak a veszélye, hogy az aktuális szoftverünknek hirtelen megszűnik a hivatalos támogatása; és most úgy néz ki, hogy az eszközök ára gyorsabban csökken, illetve a sebessége gyorsabban nő, mint ahogy a Linux sebességigénye. (Ez az ablakos rendszerről nem mindig volt elmondható.) Továbbá sokkal egyszerűbb összeállítani egy, a lehető legtöbb irodánkban használható egységes asztali rendszert, különböző kiépítésre hangolva, ami egy ekkora cégnél nagyon megkönnyíti felügyeletet.

A cikk előző részében már beszéltem a saját telepítőkészletről. Tekintve, hogy az általunk használt rendszerlenyomat jelenleg 1300 MB, CD-re írása nehézségekbe ütközik, de itt is vannak kibúvók. A jövő ezen a téren még homályos, de több megoldás is létezik (például a rendszerlenyomat DVD-re írása, ez a személyes kedvencem, vagy az általunk használt „fullextrás” rendszer karcsúsítása), és pár hónapon belül ez a rendszer (illetve az alapjain épült éppen friss lenyomat) már Afganisztánban és Pakisztánban is szolgálatot teljesíthet.

Hogy a kísérlet sikeres volt-e? A magam részéről feltétlenül annak ítélem a projektet: a Linux felkerült a gépekre, egyre többen használják, egyre használhatóbb; a bevezetőben említett PHP+MySQL követőrendszer munkába állítása után a Windows indítási lehetőségként se lesz tovább látható, és már most is elmondhatom: a mi kis Linux-rendszerünk a Windows minden szolgáltatását ki tudja váltani. A Linux alkalmas irodai feladataink ellátására.

Ha a cég alkalmazottait kérdezem, az ő válaszuk (sajnos és még) nem ilyen egyértelmű, ami nagyon fontos figyelmeztetés. Nagyon sokat kell még tenni a használhatóság és kényelem terén, hogy a „mezei” felhasználók is kényelmesen tudják használni, hogy maguktól is kedvük legyen áttérni egy Linux alapú rendszerre.

Lippai Gergő (lipilee@bekafarm.hu)
Munkaidőben rendszergazda, munkaidőn kívül Joseph Hellertől idéz mindenki füle hallatára, Monty Python-maratonokat szervez, country-dalokat énekel vagy Budapest utcáit rója.

Magas rendelkezésre állású LDAP (2. rész)

Sorozatunk második részében a Heartbeat beállításával folytatjuk magas rendelkezésre állású azonosítótelepünk kiépítését.

Rengeteg jó példát találhatunk Heartbeat-beállításokra a ***Kapcsolódó címek*** részben. Az alábbiakban láthatóak az általunk használt kiépítés idevágó beállításai, amelyek meglehetősen egyszerűek. Alapértelmezés szerint minden beállítófájl a */etc/ha.d/* könyvtárban tárolódik.
A *ha.cf* fájl, amely a géptelep számára a teljes körű megnevezéseket tartalmazza, a következőképpen fest:

```
# LejÆrati idıkeret
keepalive 2
# A keepalive-ot egy mÆsodpercre is Æll thatjuk
deadtime 10
initdead 120

# soros kapcsolat
serial  /dev/ttyS0
baud    19200

# ethernetkapcsolat
udpport 694
udp     eth1

# Øs vØg l a csom pontjaink azonos t i
# node  csom pont_neve (meg kell egyeznie
# az uname -n kimenetØvel)
node    slave5
node    slave6
```

A haresources fájl állítja be a tényleges hibatűrést. A fájl vége az érdekes:

```
slave6 192.168.10.51 slapd*
```

Ez a sor három karakterláncot tartalmaz. Elsőként a csomópont erőforrásainak elsődleges tulajdonosát, ami esetünkben a slave6 (ennek a névnek meg kell egyeznie az elsődleges csomóponton kiadott `uname -n` parancs kimenetével). A második lánc a szolgáltatásunk címe, a látszólagos IP-címünk, ami `192.168.10.51` (példánk egy magánhálózaton alapul, ezért a `192.168`-as IP-cím). Végül pedig megadjuk, hogy a szolgáltatáshoz tartozó parancsfájl a `slapd`. Ilyen nevű parancsfájlt keres majd a Heartbeat a */etc/ha.d/resource.d* és */etc/init.d/* könyvtárban.

Az indító–leállító parancsfájl

Egyszerű helyettesítés („hideg készenlét”) esetén módosítás nélkül használhatnánk a */etc/init.d/slapd* parancsfájlt. Mi azonban szeretnénk elvégezni néhány módosítást, így megalkottuk a saját `slapd` parancsfájlunkat, amit a */etc/ha.d/resource.d/* könyvtárban tárolunk (a parancsfájl a már említett `•tgz` fájlban megtalálható). A Heartbeat ezt a könyvtárat helyezi legelőre a keresési listájában, így nem kell attól félnünk, hogy a mi fájlunk helyett véletlenül a */etc/init.d/slapd* fájl futna le.

Ugyanakkor ügyeljünk arra, hogy rendszerindításkor a */etc/init.d/* könyvtárban található parancsfájl többet ne fusson le, tehát a */etc/rc.d/* könyvtárból távolíts el minden `S*slapd` fájlt.
Az indító parancsfájlban két különböző indulási felállásnak megfelelő beállítás is helyet kapott, így biztosítva számunkra, hogy a kiszolgálót akár mesterként, akár segédként elindíthassuk. Amikor a parancsfájl elindul, első lépésben a `slapd` összes futó példányát leállítja. Ezt követően megvizsgálja, hogy az elsődleges és a másodlagos csomópont egyaránt fut-e. Ha igen, akkor a `slapd`-t mesterként (a főkiszolgálón) vagy segédként (a másodkiszolgálón) indítjuk. Ha a két gép közül csak az egyik él, a szolgáltatást mesterként indítjuk rajta. Ez azért is fontos, mert a látszólagos IP-cím mindig a mesterre mutat.
Hogy ezt megtehessük, tudnunk kell, milyen csomóponton fut le a parancsfájl. Ha a főkiszolgálón fut, a másodkiszolgáló állapotát szintén ismernünk kell. Az érdekes rész a parancsfájl „start” ágában található. Mivel a Heartbeat beállításánál megjelöltünk egy elsődleges csomópontot, ha a `test_start()` függvény lefut, a főkiszolgálón vagyunk. (Mivel a Heartbeat a */etc/init.d/* könyvtárban lévő parancsfájlok szerkezetét használja, minden fájlt a `start|stop|restart` kapcsolók egyikével hívnak.)
A Heartbeat a parancsfájlok meghívásakor egész sor környezeti változót állít be. Minket azonban elsősorban a `HA_CURHOST` érdekel, melynek értéke `slave6`. A `HA_CURHOST` változót használhatjuk fel arra, hogy megállapítsuk, a parancsfájl éppen a főkiszolgálón (`slave6`) vagy a másodgépen (ekkor a `HA_CURHOST` értéke `slave5`) fut-e.
Most pedig meg kell tudnunk a másik csomópont állapotát, így megkérdezzük a Heartbeatet. Ehhez készítettünk egy egyszerű programot, amelyhez felhasználtuk az `api_test.c` állományt. (Az `api_test.c` még számos más feladatot ellát, a számunkra szükségtelen dolgokat eltávolítottuk.) A programot `other_state` néven fordítottuk le, majd bemásoltuk a */etc/ha.d/resource.d/* könyvtárba.

Az indítófájl ellenőrzése

Most már mindkét kiszolgálón elindíthatjuk a Heartbeatet. A Heartbeat leírása tartalmaz pár példát az alapbeállítások ellenőrzéséhez, így most nem ismételjük meg őket. Két Heartbeat két közvetítővel, ami egy hálózati meg egy soros kapcsolatból áll, vagyis összesen hat Heartbeat-folyamatnak kell futnia. A hibatűrés ellenőrzésére több kísérletet végezhetünk el.
A kiszolgálók ellenőrzéséhez egy egyszerű KDE-alkalmazást készítettünk, ami a kiszolgálókat lekérdezi kapcsolataik állapotáról. Egy valódi ügyfélprogram csak a látszólagos IP-címet kérdezné le, de hogy a működését bemutathassuk, mi mind a három IP-címet lekérdezzük. A kísérlet során óránként 10 000 kérést küldünk (*1. kép*).
Az S6 az LDAP-főkiszolgáló, és ahogyan az a *2. képen* látható, az S5 a másodkiszolgáló. Az alsó „gép” a látszólagos IP. Alaphelyzetben az S5 és az S6 is zöld színű, ami azt jelenti, hogy a kérések sikeresek.

1. kép Az S6 (a főkiszolgáló) és az S5 (a másodkiszolgáló) egyaránt fut

2. kép A főkiszolgáló leállását követően a szolgáltatásokat másodkiszolgáló veszi át

A kísérletet úgy kezdjük, hogy a főkiszolgálón leállítjuk a `heartbeat` folyamatot. Ekkor a másodkiszolgáló 10 másodperces kiesési idő után átveszi az erőforrásokat – erről a naplóból bizonyosodhatunk meg. Az átvétel előtt az indítófájl még további két másodpercet várakozik (2. *kép*). Ha a főkiszolgáló leáll, a látszólagos IP-t a másodlagos kiszolgáló veszi át, a 2. képen szereplőknek megfelelően. Az S5 és a látszólagos cím zöldet mutat, az S6-os gép viszont elérhetetlen, így az ahhoz tartozó kijelző piros. A géptelep újraindítását követően újabb hibát idéztünk elő: egy laza mozdulattal kihúztuk a főkiszolgáló tápkábelét. A másodkiszolgáló tíz másodperces várakozás után ebben az esetben is átvette a szolgáltatást.
Végül a kapcsolattartást zavartuk meg: eltávolítottuk a két kiszolgáló közötti soros és hálózati kábelt. A kapcsolattartás ilyen mérvű megszakadása mindkét kiszolgálót arra készteti, hogy mesterként próbáljanak meg viselkedni. Ez az „agyhasadás" (split-brain) jellemző esete. Ez a példa jól szemlélteti, miért szükséges a gépek közti biztonságos szívhangkapcsolat kiépítése. Egy osztott tárolórendszerben a tárolóeszközhöz való közös kapcsolat is alapját képezheti a szívhang működésének, mely csökkenti az „agyhasadás"-helyzetek előfordulását.
Erre a helyzetre az időkorlátok megválasztásánál tekintettel kell lennünk. Ha az időkorlát túl rövid, egy leterhelt rendszer azt a látszatot keltheti, hogy kiesett a szolgáltatásból, amivel agyhasadást idéznénk elő. A témával kapcsolatban további érdekességeket találsz a Linux-HA Gyakori kérdések részében.

Helyreállítás egy kiesést követően

Ha az adatbázisban változások következnek be, amíg a főkiszolgáló üzemen kívül van, akkor – mielőtt a főkiszolgálót újraindítanánk – az LDAP-adatbázisokat egyeztetni kell. Erre két lehetőség létezik. Ha a szolgáltatás megszakítható, az adatbázist kézzel átmásolhatjuk, miután a helyettesítő LDAP-kiszolgálót is leállítottuk. (Az adatfájlok alaphelyzetben a */usr/local/var/* könyvtárban találhatók.)
Ugyanakkor az OpenLDAP többszörözőjét is használhatod, hogy anélkül állítsd vissza az adatbázist, hogy a szolgáltatást meg kellene szakítani. Ehhez először segédként indítsd el a leállított főkiszolgálót, majd a jelenlegi mesteren indítsd el a `slurpd` programot. Így a helyettesítés alatt keletkezett változások szépen áttöltődnek az eredeti főkiszolgálóra. Ezt követően az eredeti főkiszolgálón állítsd le az LDAP-szolgáltatást, és indítsd el a Heartbeatet, ami a szokásos üzembe állítja vissza a rendszert, és a főkiszolgáló visszaveszi az LDAP-szolgáltatás mesterének a szerepét.

Összegzés

Ez a cikk azt mutatta be, hogyan készíthetünk magas rendelkezésre állású hálózati szolgáltatásokat nyílt forrású eszközök segítségével. Az olyan hálózati szolgáltatások, mint amilyen az LDAP is, ritkán igényelnek nagy gépeket. A két kiszolgálógép használatának és telepbe kötésének járuléka a magasabb rendelkezésre állás és a nagyobb megbízhatóság. Az írásunkban bemutatott rendszer folyamatosan működött, mindegyik leállás 15 másodperc alatti volt. Természetesen a rendszerek terhelését figyelembe véve finomhangolással ez az idő igény szerint tovább csökkenthető.

Köszönetnyilvánítás

Köszönet ***Alan Robertson***-nak, az IBM Linux Technology Center munkatársának a sok értékes tanácsért és megjegyzésért.

Figyelmeztetés

A leírt kísérlet eredményei laboratóriumi eredmények, vagyis laboratóriumi körülmények között keletkeztek. Más felhasználók gépein az eredmények rengeteg dologtól függhetnek, beleértve a gépek terheltségét és beállításait is. A cikkben szereplő adatokat ennek fényében kell értelmezni.

A leírtakkal kapcsolatban semmilyen garanciát vagy felelősséget nem vállalunk! A leírtakat mindenki a saját felelősségére használja!

Linux Journal 2002. december, 104. szám

Cliff White (cliffw@osdl.org)
Az OSDL műszaki munkatársa (➲ http://www.osdl.org). 1989 óta dolgozik különböző Unix- és Linux-változatokkal. Hogy biztos legyen a dolgában, minden reggel azonosítja önmagát.

Jay D. Allen (allen5@us.ibm.com)
Nappal programfejlesztőként a legújabb IT-megoldásokon dolgozik az IBM Linux for Service Providers Labnél (LSPL), ahol Linuxot használ Intelen és RS/6000-en. Éjszakánként elmaradott megoldásokkal foglalatoskodik, főként DEC PDP-11-gyel és egyéb régiségekkel.

KAPCSOLÓDÓ CÍMEK

Minden parancsfájl elérhető a 44. CD Magazin/HALDAP könyvtárában és a Linux Journal ftp-kiszolgálóján, az ➲ ftp.ssc.com/pub/lj/listings/issue104/5505.tgz címen.
A Heartbeat beállításai
➲ http://www.linux-ha.org/download/GettingStarted.html
Linux-HA honlap ➲ http:// www.linux-ha.org
Linux Virtual Server projekt
➲ http://www.LinuxVirtualServer.org
OpenLDAP honlap ➲ http://www.openldap.org
Ultramonkey ➲ http://ultramonkey.sourceforge.net

Használjunk IPv6-ot a Freenet6-tal!

Hogyan állítsunk fel IPv4-be ágyazott IPv6-alagutat, és szolgáltassunk IPv6-hálózatot átjáró mögötti gépeinknek.

Az IPv6, jelenlegi internetprotokollunk, az IPv4 utódja. Számos új szolgáltatást kínál, beleértve a jelentősen megnövelt címtartományt (128-bites címek az eddigi gyenge 32-bites IPv4 helyett), könnyebb, önműködő beállításokat és jobb titkosítástámogatást. A Debian Projekt jó munkát végzett terjesztésének IPv6-felkészítése terén. Írásunkban azt mutatjuk be, hogyan lehet ***IPv4-be ágyazott IPv6-alagutat*** (IPv6-over-IPv4 tunnel) alagutat létrehozni, és IPv6-hálózatot biztosítani az átjáró mögötti gépeknek az ingyenes Freenet6 alagútszolgáltatás és a `radvd` segítségével. A cikkben feltételezni fogom, hogy Debian Woodyt (3.0) vagy fejlesztői (unstable, SID) rendszert használunk, illetve tisztában vagyunk azzal, hogyan lehet csomagokat telepíteni az `apt`-vel. Azt is feltételezem, hogy valamennyi alapismerettel rendelkezünk már az IPv6 terén, például tudjuk, hogyan néz ki egy cím. Átjárónkon szükségünk lesz egy nyilvánosan elérhető IPv4-címre. Az átjáró rendszermagjában és az összes bekapcsolni kívánt gépen meg kell lennie az IPv6-támogatásnak – a Debian rendszermagja alapértelmezés szerint támogatja az IPv6-ot. Ha nem vagyunk benne biztosak, hogy a miénk támogatja-e vagy sem, nézzük meg a */proc/sys/net/ipv6* könyvtárat, illetve pillantsunk rá, hogy az `ifconfig` kimenete tartalmaz-e olyan IPv6-címeket, mint például fe80::24f:49ff:fe07:2552. Bizonyosodjunk meg afelől, hogy tűzfalunk nem szűri-e az IPv6-alagút a csomagjait. Az IP 41-es protokoll (ipv6 ipv4 fölött) egyik irányban sem lehet tiltva.

Az IPv6-alagút működés képtelen lesz, ha NAT útválasztó mögött bújunk meg, de jó hír, hogy nincs szükségünk szolgáltató által adott állandó IPv4-címre ahhoz, hogy IPv6-címünk lehessen. Először is telepítsük a Freenet6 ügyfélprogramot az `apt`-vel. A Debian csomag neve `freenet6`, és a Woodyban vagy a frissebb kiadásokban már megtalálható. RPM- és forráscsomagok is elérhetők a ➲ http://www.freenet6.net/download.shtml címen. Ha feltelepítettük, a Freenet6 a már létező IPv4-címünk alapján önműködően lekéri az IPv6-címet; további beállításokra nincs is szükség.

A */etc/init.d/freenet6* parancsfájl egyszerűen csak elindítja az alagutat felépítő programot, név szerint a ***Tunnel Setup Protocol Client***-et (`tscp`). Ettől kezdve, ha az IPv4-címünk megváltozik, IPv6-címünk is módosulni fog. Rendben, tehát a legtöbb rendszeren a Freenet6 telepítése után a gépen máris minden működni fog. Ugyanakkor esetleg további lépéseket is tehetünk – felállíthatjuk a saját IPv6-alhálózatunkat. Ehhez szükségünk lesz egy felhasználói azonosítóra.

Keressük fel a ➲ http://www.freenet6.net-et, és kattintsunk a ***Create your account*** (azonosító készítése) szövegre, amelyet a bal kéz felőli menü ***TSP Server*** részében találunk. Nevünk és levélcímünk megadása után a rendszer postázza nekünk a kért felhasználói azonosítót és jelszót.

Kedvenc szövegszerkesztőnkkel nyissuk meg a */etc/freenet6/tspc.conf* fájlt. Ez a beállításfájl vezérli a `tspc`-t, alapértelmezés szerint névtelen (anonim) kapcsolatot hozva létre. Keressük ki az ***user ID*** és ***password*** sorokat, majd változtassuk meg az értéküket azokra, amelyeket a levélben kaptunk. Indítsuk újra a Freenet6-ot (futtassuk a `/etc/init.d/freenet6 restart` parancsot), és máris van egy tökéletesen működő IPv6-beállításunk, egyetlen IPv6-címmel. Ha csak ennyire van szükségünk, itt akár abba is hagyhatjuk az olvasást.

Feltételezve, hogy ennél többre vágyunk, a következő lépés saját /48-as előtagunk lekérése. Ezt ismét a ***tspc.conf*** szerkesztésével tehetjük meg. A következő sorokat adjuk a ***tspc.conf*** fájl végéhez:

```
host_type=router
prefixlen=48
if_prefix=eth0
```

Az `if_prefix` kapcsoló a belső hálózatunkhoz rendelt csatolót hivatott beállítani. A Freenet6 önműködően elindítja e csatolót és a Freenet6 alagút között az IPv6-átirányítást. A legtöbb felállásnál az `eth0` lesz a jó választás, de természetesen igény szerint megváltoztathatjuk. E cikk további részeiben a belső hálózat csatolójára `eth0` néven fogunk hivatkozni. Ha kiépítésünk más csatolót használ, használjuk azt helyette. Ezen a ponton újraindíthatjuk a Freenet6-ot (a `/etc/init.d/freenet6 restart` paranccsal töltve be az új beállítást), mert további változtatásokat már nem tervezünk benne. Ha újraindítás után lefuttatjuk az `ifconfig`-ot, egy új IPv6-címet láthatunk az `eth0`-n XXX:XXXX:XXXX:1::1/64 formátumban. A Freenet6 önműködően adta az `eth0`-nak ezt az új címet a /48-as előtagunk alapján. A XXXX:XXXX:XXXX rész a /48-as előtagunk hálózati része. Ha akarjuk, állandósíthatjuk is ezt a címet, ha hozzáadjuk a */etc/network/interfaces* `eth0` részéhez. Ez lehetővé teszi, hogy más gépek még akkor is megtalálják az átjárónkat, ha a Freenet6 program nem fut. Ehhez keressük meg az `eth0` szakaszt, ami valahogy így nézhet ki:

```
iface eth0 inet static
        address 10.1.1.1
        netmask 255.255.255.0
        network 10.1.1.0
        broadcast 10.1.1.255
```

Adjuk a következő sorokat (a 3ffe:0b80:083b:1::1 címet a Freenet6-tól kapott címünkkel helyettesítjük, a /64 hálózati maszk elhagyásával) a fájlhoz:

```
iface eth0 inet6 static
        address 3ffe:0b80:083b:1::1
        netmask 64
```

Próbáljuk is ki az `ifdown eth0` és `ifup eth0` parancsokkal. Egy trükk: ha ezt távolról szeretnénk megtenni, és az `eth0` az a csatoló, ahonnan bejöttünk, elkerülhetjük a kapcsolat elvesztését és a géphez való sétálást, ha a két parancsot pontosvesszővel kapcsoljuk össze:

```
# ifdown eth0 ; ifup eth0
```

Merész szerzőnk is pontosan ezt szokta tenni SSH-kapcsolatán keresztül (próbagépem a földszinten, én viszont az emeleten vagyok).
Ha eddig minden jól ment, nekiláthatunk a `radvd` beállításának. Az IPv6 sokkal több önműködő beállítást tesz lehetővé, mint az IPv4. Ahhoz, hogy ez az önműködő beállítás működjön, a hálózat útválasztóinak válaszolnia kell a gépek önműködést beállító kéréseire. Linux alatt ezt a programot radvd-nek (Router ADVertisement Daemon) hívják. A tulajdonképpeni beállítás úgy működik, hogy a gépeknek a 64-bites hálózati cím és az ethernetkártya alkatrészcímek alapján oszt ki új IPv6-címeket. A gépek azt is megkapják, minek kell lennie az alapértelmezett átjárójuknak. Akárcsak a Freenet6 esetében, a radvd-hez is létezik Debian-csomag.
Miután feltelepítettük a `radvd`-t, a */etc/radvd.conf* fájlt át kell szerkesztenünk. A Debian alapértelmezett beállítása semmit sem csinál, így az egészet a következőkkel helyettesítsük:

```
interface eth0
{
   AdvSendAdvert on;
   prefix 3ffe:b80:840:1::/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
```

Az előtag értékét az `eth0` hálózati cím és maszk értékekre állítjuk be. Ha például az `eth0`-hoz rendelt IPv6-cím Freenet6 alatt 3ffe:b80:83b:1::1/64 volt, az előtagot 3ffe:b80:83b:1::/64-ra kell állítani, ami azonos az IPv6-címmel, eltekintve a cím gazdagéprészétől (azaz a :1-től). Ha ezzel megvagyunk, indítsuk újra a `radvd`-t.
Újraindítás után próbáljuk meg lefuttatni a belső hálózatunkhoz kapcsolt IPv6-os gépeken az `ifconfig`-ot. Ha minden jól megy, a belső hálózathoz csatlakoztatott felületen egy új IPv6-címet kell találnunk. Amennyiben a fenti előtagot használtuk, az egyik ilyen cím ez lehetne: 3ffe:b80:840:1:24f:49ff:fe07:2552/64. Amennyiben új címek nem jelennének meg, nézzük meg, nem találunk-e hibajelentéseket a `radvd`-t futtató gép */var/log/syslog* naplójában. A `radvd` a képernyőre nem naplóz.
IPv6-kapcsolatunk kipróbálásához először is az átjárón és a csatlakoztatott gépeken próbáljunk a `ping6` programmal megpingelni valamit. Előfordulhat, hogy a `ping6` még nincs telepítve. Amennyiben ez lenne a helyzet, telepítsük az `iputils-ping` csomagot. A `ping6`-ot kipróbálhatjuk a ➲ http://www.6bone.net és a ➲ http://www.kame.net címeken. Ha nem működne, még egyszer ellenőrizzük, tűzfalunk hátha mégiscsak tiltja az IPv6-csomagokat. Ha működik, próbáljunk meg fellépni a ➲ http://www.kame.net-re Lynx-szel, Mozillával vagy Konquerorral. Ezek a böngészők a Debian próbaváltozatában már rendelkeznek IPv6-támogatással. Amennyiben minden működik, egy táncoló „kame”-t fogunk a lap tetején látni, illetve egy apró üzenetet olvashatunk a lap alján. Ha itt azt olvassuk, hogy még mindig IPv4-et használunk, de a `ping6` próba jól működött, ellenőrizzük, hogy a legfrissebb változatú böngészőt használjuk-e.
Az IPv6-alagút létrehozása egyben biztonsági lyukat jelenthet. A 2.4.x rendszermagokban található IP Tables lehetővé teszi IPv6-tűzfalak készítését, ennek ismertetése azonban már meghaladja cikkünk kereteit. Erősen ajánlott a lehetséges biztonsági rések végiggondolása. Az egyik módszer, amivel megtudhatjuk, hogy milyen IPv6-szolgáltatások nyíltak meg, ha lefuttatjuk a `netstat -l -A inet6` parancsot. Itt az összes nyitott IPv6-foglalatot megtekinthetjük. Ha vannak nyitottak, győződjünk meg róla, hogy rendben van-e, ha nyilvánosan is elérhetők.
Gratulálok! A következő nemzedékbeli Interneten lógsz. Most, hogy már van alapvető IPv6 szolgáltatásunk, érdemes foglalkozni az egyéb szolgáltatások, például a webböngészés és a levelezés IPv6-osításával. A Debian IPv6 Projekt jó kiindulási hely, ha a Debian-csomagok foltozott változatait keressük. Bár a legtöbb ügyfélprogram már eleve IPv6-megfelelő, azért még elég sok kiszolgáló és démon létezik, például az `inetd`, az Apache és a többi, amelyek még nem kerültek be a fővonalbeli Debian-változatokba. Ezeket a csomagokat egyszerű telepíteni; csak adjunk pár sort APT beállításfájlunkhoz, majd futtassuk a `dist-upgrade` parancsot. Akit érdekel, a ➲ http://people.debian.org/~csmall/ipv6 helyen nézhet körül.

Linux Journal 2003. január, 105. szám

Peter Todd (pete@petertodd.ca)
14 éves kora óta használ Linuxot. Amikor éppen nem a legfrissebb hálózati protokollokkal bütyköl, a Woburn Főiskolán található, ahol fazekassággal és filmművészeti projektekkel foglalkozik.
A ➲ http://petertodd.ca címen saját IPv4 és IPv6 alapú honlapjára bukkanhatunk.

KAPCSOLÓDÓ CÍMEK

Debian IPv6 Project
➲ http://people.debian.org/~csmall/ipv6
Freenet6 Tunnel Service ➲ http://www.freenet6.net
IPv6 információs oldalak ➲ http://www.ipv6.org
Peter Bieringer linuxos IPv6 oldala
➲ http://www.bieringer.de/linux/IPv6

IPv6-szavú Apache

Segédlet az IPv6 szerinti HTTP-kérések Apache alatti beállításához, és néhány előzetes próbaeredmény.

Az IP6 változat (IPv6) az Internet Protokoll legfrissebb változata, amelyet az IETF tervezett az IPv4 kiváltására. E cikkben arról írunk, hogyan készíthetjük fel az Apache webkiszolgálót az IPv6 alatti futásra.

Az Apache webkiszolgáló

A Netcraft webkiszolgáló-felmérése szerint 1996 áprilisától kezdve az Internet legnépszerűbb webkiszolgálója az Apache. Jelenleg a webkiszolgálók több mint 56 százaléka Apache-ot futtat. Ezek a számok egyáltalán nem meglepőek, ismerve az Apache különféle felületek közti hordozhatóságát, megbízhatóságát, erőteljességét, beállíthatóságát, és nem utolsó sorban tudván, hogy ingyenes és jó leírással rendelkezik.

Az Apache 1.3 nagy teljesítményű webkiszolgálóként már bizonyított. Csakhogy a webkiszolgálókkal szemben támasztott követelmények növekednek; a jelenkor igényei között olyan új szolgáltatások szerepelnek, mint nagyobb megbízhatóság, biztonság, teljesítmény és méretezhetőség. Válaszul az Apache folytatta előrenyomulását, és 2.0-s változatában megpróbálja kielégíteni az új elvárásokat: új és fejlettebb képességekkel felvértezett, még erőteljesebb és gyorsabb webkiszolgálót ígér.

Az Apache 2.0 számos teljesítménynövelő fejlesztést mutat be. Nekünk azonban ebben a cikkben most csak egyetlen ilyen képesség lesz fontos, mégpedig az IPv6-támogatás. A 2.0-s változattól kezdve, amennyiben az Apache-ot IPv6-támogatással rendelkező rendszeren futtatjuk, az Apache alapértelmezés szerint hallgatni fog az IPv6-foglalatokon (socket). Ezen felül a `Listen`, `NameVirtualHost` és `VirtualHost` beállítások szintén támogatják az IPv6 formátumú címeket.

A következő részekben bemutatjuk, hogyan illesszük az IPv6-támogatást a Linux-rendszermagba, majd leírjuk, miként telepítsük és futtassuk a legfrissebb Apache-változatot IPv6-támogatással. Végül lefuttatunk néhány teljesítménymérő tesztet, hogy összehasonlíthassuk ugyanannak a kiszolgálónak a teljesítményét IPv4 és IPv6 alatt.

Rendszermagszintű IPv6-támogatás

Ebben a részben röviden ismertetjük, hogyan kell engedélyezni a Linux-rendszermag IPv6 szolgáltatását, amely előfeltétele az IPv6 HTTP-kérelmek fogadásának. Elképzelhető, hogy rendszermagunk már eleve rendelkezik IPv6-támogatással (erről bővebben lásd még a 24–25. oldalt). A beillesztéshez először is töltsük le a legfrissebb üzembiztos Linux-rendszermagot, majd csomagoljuk ki. A kipróbáláshoz mi a 2.4.8-as változatot töltöttük le a kernel.org-ról. (A jelenlegi üzembiztos rendszermag a 2.4.20-as – a ford.)

Beállítottuk a rendszermagot, hogy támogassa az IPv6-ot. Két lehetőséget kell engedélyeznünk: az egyik a ***Code Maturity Level*** részben a ***Prompt for development and/or incomplete code/drivers*** pont. A másik a ***Networking Options*** részben az ***IPv6 Protocol (EXPERIMENTAL)*** lehetőség, amelyet engedélyeznünk kell. Ízlés szerint kiválaszthatjuk, hogy az IPv6-támogatást a rendszermagba vagy külön modulba szeretnénk-e helyezni. Ezt követően fordítsuk le, majd a szokásos módon telepítsük a rendszermagot és a modulokat, és a gépet IPv6-támogatással indítsuk újra.

Az Apache letöltése és telepítése

Most, hogy rendszermagunk már támogatja az IPv6 protokollt, készen állunk az Apache telepítésére és IPv6-támogatású futtatására. Elsőként töltsük le a legfrissebb Apache-terjesztést a ➲ http://www.apache.org/dist/httpd helyről a ***/tmp*** könyvtárba. E bemutató céljára az Apache 2.0.16-ot fogjuk használni; természetesen ugyanezek a lépések az újabb változatok esetében is érvényesek. Bontsuk ki a forrást:

```
# cd /tmp
# tar xzvf httpd-2_0_16-beta.tar.gz
```

ami létrehozza a forráskódot tartalmazó ***httpd-2_0_16*** nevű könyvtárat. Az Apache-t hozzá kell igazítanunk a saját felületünkhöz és különleges igényeinkhez. Ezt a `configure` parancsfájl segítségével tehetjük meg, amelyet a telepítés gyökérkönyvtárában találnunk meg. Belépünk a ***httpd-2_0_16*** könyvtárba, majd a parancssorba begépeljük a `./configure` utasítást. Ha kíváncsiak vagyunk, milyen kapcsolókat adhatunk át a beállító parancsfájlnak, írjuk be a `./configure -help` utasítást. Ahhoz, hogy az alapértelmezett kapcsolók megváltoztathatók legyenek, a beállító parancsfájl különféle változókat és parancssori kapcsolókat egyaránt elfogad.

Az egyik ilyen kapcsoló az Apache telepítési helyének előtagja. Alapértelmezés szerint az Apache a ***/usr/local/apache*** könyvtárba kerül. Ha az Apache-t inkább a ***/usr/local/apache-2_0_16*** könyvtárba tennénk, akkor a következő parancsot írjuk be:

```
# ./configure --prefix=/usr/local/apache-2_0_16
```

Ugyanez vonatkozik a többi kapcsolóra is. A `configure` futás közben néhány másodperc alatt végigvizsgálja, hogy milyen képességek érhetők el a rendszerünkön, majd elkészíti a kiszolgáló felépítéséhez szükséges `Makefile`-okat. A beállítási parancsfájl lefuttatását követően a `make && make install` parancs kiadásával elkészíthetjük az Apache csomagot alkotó különböző részeket. A parancs lefordítja az Apache-ot, majd telepíti a ***/usr/local/apache-2_0_16*** könyvtárba. A következő lépés az Apache HTTP-kiszolgáló testreszabása a ***/usr/local/apache-2_0_16/conf*** könyvtár alatt található ***httpd.conf*** fájl szerkesztésével. Amennyiben más előtagot adtunk meg, természetesen az ***előtag/conf/*** könyvtárban kell keresnünk:

```
# vi elıtag/conf/httpd.conf
```

A mindenképpen átszerkesztendő beállítási kapcsolók közé tartozik a `ServerAdmin`, ahová a saját levélcímünket kell beírni (így figyelmeztetést fogunk kapni arról, ha a kiszolgálónak gondjai lennének); a `ServerName`, azaz a kiszolgáló

A teljesítménymérés eredményei

	Apache 1.3.19-cel IPv4	Apache 1.3.19-cel IPv6	Apache 2.0.16-tal IPv4	Apache 2.0.16-tal IPv6
Kiszolgálóprogram	Apache/1.3.19	Apache/1.3.19	Apache/2.0	Apache/2.0
Kiszolgáló neve	node01.lmc.ericsson.se	node01.ip6.lmc.ericsson.se	node02.lmc.ericsson.se	node02.ip6.lmc.ericsson.se
Kérések száma másodpercenként	109.28	105.78	104.94	101.37
Készenléti idő	9,15 ms	9,45 ms	9,53 ms	9,87 ms
Átviteli sebesség	1275,38 KB/s fogadott	1234,50 KB/s fogadott	627,35 KB/s fogadott	605,98 KB/s fogadott

IP-je vagy neve; és a `Port`, vagyis az a kapu, amire a kiszolgálónak hallgatnia kell.
A ***httpd.conf*** sok magyarázatot is tartalmaz és elég könnyű olvasni, értelmezni és testreszabni. Ha mégis további részletekre vagyunk kíváncsiak a beállításokkal kapcsolatban, elolvashatjuk az Apache-kézikönyvet saját gépünk Apache telepítési könyvtárának ***docs/manual/*** alkönyvtárában vagy a hálózat ➲ http://***httpd.apache.org/docs*** címén.

Kiszolgáló indítása

Sikeresen lefordítottuk, telepítettük és testreszabtuk Apache rendszerünket. Mint korábban említettük, az IPv6-támogatás ma már része az Apache forráskódjának, így a beindításához nincs szükség további különleges lépésekre. A kiszolgáló indítását az Apache-vezérlést egyszerű parancssoros felülettel lehetővé tevő ***Apache Control Script*** segítségével végezhetjük el. Ezáltal elindíthatjuk, leállíthatjuk, újraindíthatjuk a kiszolgálót, szemmel tarthatjuk az állapotát, illetve ellenőrizhetjük a beállítások írásmódjának helyességét. Ennek megfelelően a kiszolgáló indításához az `apachectl start` parancsot kell kiadnunk, és a dokumentumokat máris lekérhetjük a ➲ http://***kiszolgálónév*** vagy a http://localhost/ címen.

Dokumentumszolgáltatás IPv6 alatt

A 128 bites címek eljövetelével egyidejűleg a hosszú IP-címek gépelésének kellemetlensége is megérkezik. Az IPv6-címek helyes írásmódja nyolc, kettősponttal elválasztott, négy hexadecimális karaktert tartalmazó mezőből áll, ami összesen 128 bitet jelent. Emlékezzünk rá, hogy az URL-ekben a kapuszámokat általában kettős ponttal lehet a címhez fűzni, tehát a következőképpen: 149.76.14.14:80. Minthogy az IPv6-címek eleve kettőspontot használnak magában a címben is, az IPv6 URL-ekben a címeket szögletes zárójelbe kell tenni, és csak ezt követően lehet kettős ponttal hozzáfűzni a kapu számát, valahogy így:

```
[3ffe:200:8:1000:250:bbff:fe00:25]:80.
```

Nem minden webböngésző képes értelmezni az IPv6-címeket. A Netscape Navigator 6.x és a Mozilla azonban két kipróbált böngésző, amelyek támogatják az IPv6-címzést. Úgy gondoljuk, hogy bár IPv6-támogatottságuk még nem teljesen kiforrott (hiszen néha lefagynak, és a böngészőfolyamatot ki kell lőni vagy újra kell indítani), mindkettő sikeresen képes elérni a másodneveket (aliased) gépneveket. A másodnevek olyan gépnevek, amelyeket valamilyen egyedi IP-címhez rendeltünk a ***/etc/hosts*** fájlban. Például másodnevet rendelhetünk egy IPv6-címhez, ha a ***/etc/hosts*** fájlba a következőket írjuk:

```
::1                         node02-v6-localhost
3ffe:200:8:1000:250:bbff:fe00:25    node02-v6
```

Amikor elindítjuk a Netscape-et vagy a Mozillát, címként begépeljük a megadott álnevet, majd ezekkel a másodnevekkel érjük el az IPv6 alatti weblapokat, a fenti példánál maradva például a ➲ http://node2-v6/ vagy a ➲ http://node2-v6-localhost/ lapokat.

Teljesítménymérési eredmények

Az Apache teljesítményének méréséhez a fenti lépéseket követve feltelepítettük a legfrissebb rendszermagot IPv6-támogatással, majd felraktuk az Apache 2.0.16-as változatát. A gép egy 256 MB memóriával rendelkező 1U Celeron 500 MHz szekrénybe szerelhető egység volt, a rendszer Red Hat 7.0 alatt működött. A teljesítménymméréshez az Apache webkiszolgálóval érkező ingyenes ApacheBench programot használtuk.
Két tesztet futtattunk le: Apache 2.0.16 IPv4 alatt, majd Apache 2.0.16 IPv6 alatt. Az összehasonlítás kedvéért ugyanezt a próbaprogramot lefuttattuk egy azonos beállításokkal rendelkező gépen is, amelyen Apache 1.3.19 futott, így összehasonlíthattuk a két változat IPv6-támogatásának sebességét. Hogy az Apache 1.3.19 alatt is elindíthassuk az IPv6-ot, a Kame Projekt honlapjáról letöltöttük az IPv6 Apache-foltot, majd az 1.3.19 forrásfájára alkalmaztuk. Ezt követően lefuttattuk a `configure` parancsfájlt:

```
# ./configure --enable-rule=INET6
```

és engedélyeztük az INET6 lehetőséget. Végül elvégeztük a `make && make install` parancsot, ami lefordította és telepítette a Kame IPv6-támogatással kiegészített Apache 1.3.19 változatot.
A *táblázatban* láthatjuk a próbaeredményeket. Minden teljesítménypróba 1000 darab 1-es konkurenciaszinten lefuttatott kérés eredményeként adódott. Egyetlen hibás kérelem vagy írási hiba sem jelentkezett.
Az eredménnyel kapcsolatban néhány megjegyzéssel kell élnünk. Az Apache 1.3.19 másodpercenként több kérelmet volt képes kiszolgálni, mint az Apache 2.0.16. Az Apache általunk kipróbált változatában IPv6 alatt kevesebb kérelmet értünk el másodpercenként, mint az IPv4 használatakor; ennek egyik oka az lehet, hogy az Apache-hoz adott IPv6-kód nincsen módszeresen végigellenőrizve és javítva. Másrészről, ha az Apache 1.3.19 átviteli sebességét tekintjük, azt tapasztalhatjuk, hogy jelentősen magasabb a Apache 2.0.16 teljesítményénél. Ez még további vizsgálatokat igényel.
Az Ericsson Open Architecture Research Labnél mostanában építettünk fel egy teljesítménymérő környezetet, amelyben Linux-telepek és alkalmazáskiszolgálók képességeit vizsgálhatjuk (ide értve az Apache, a Tomcat és a Jigsaw webkiszolgálókat). Csakhogy a környezet és a próbák IPv4-hez, és nem IPv6-hoz készültek. Azt tervezzük, hogy a környezetet IPv6 HTTP-kérelmek kipróbálására építjük át, hogy az Apache (és más webkiszolgálók) teljesítményét komoly IPv6-terhelés

mellett is meghatározhassuk. Jelenleg összesen 100 darab 1U méretű egységgel rendelkezünk (Celeron 500 és Pentium III-as gépek vegyesen, 256 MB és 512 MB memóriával), amelyek hamarosan egyetlen csomópontra menő IPv6-forgalmat hoznak majd létre, ahol összegyűjtjük és megszerkesztjük az eredményeket. Amint befejeztük a munkát, az eredményeket közzétesszük. Ez az összegzés már sokkal többet fog mondani, mint a cikkben szerepeltetett előzetes próbák eredményei.

Összegzés

Az IPv6 kezd valósággá válni. A következő néhány évben azonban alkalmazáskiszolgálóinkon képesnek kell lennünk az IPv4 és az IPv6 protokollok együttes támogatására, még mielőtt a teljes IPv6-átállás megtörténhetne. Az Apache fejlesztői ügyeltek erre, és az IPv6-támogatást már be is építették az Apache kódjába. Előzőleg az IPv6 csak letölthető folt formájában volt elérhető (többek között a Kame Projekt honlapjáról), amit aztán a forrásfába kellett illeszteni.

Mint azt a teszteredmények sugallják, az IPv6 alatti dokumentumszolgáltatás valamivel lassabb lesz (lekérés/másodperc értelemben), mint amilyen az IPv4 alatt volt; ami érthető is, hiszen az IPv6-támogatás még gyerekcipőben jár. A fejlesztés előrehaladtával az Apache is valószínűleg eléri, majd meghaladja az IPv4 alatt elért teljesítményt, hogy továbbra is a világ legkedveltebb webkiszolgálója maradhasson.

Köszönetnyilvánítás

Köszönet az Ericsson Researchnek (Open Architecture Research Lab) linuxos és nyílt forrású munkáink támogatásáért, illetve hogy engedélyezték a cikk megjelenését.

Linux Journal 2003. január, 105. szám

KAPCSOLÓDÓ CÍMEK

Apache webkiszolgáló ➲ http://www.apache.org
ARIES Projekt ➲ http://www.risq.ericsson.ca
Kame Projekt ➲ http://www.kame.net
Linux-rendszermag ➲ http://www.kernel.org
Mozilla ➲ http://www.mozilla.org
Netscape ➲ http://www.netscape.com
Red Hat ➲ http://www.redhat.com

Ibrahim Haddad (Ibrahim.Haddad@Ericsson.com)
A montreali Ericsson Research Open Architecture Lab kutatója, ahol elsődleges feladata a hírközlési osztályú kiszolgáló csomópontok kutatása valós idejű, teljesen IP alapú hálózatok területén. Jelenleg DrSc-jelölt a Concordia Egyetemen.

David Gordon (davidgordonca@yahoo.ca)
Most fejezte be együttműködését a Kanadai Ericsson Researchs-szel, ahol a közel telekommunikációs szintű Linux-telepek IPv6-támogatásán dolgozó csapat tagja volt. Jelenleg Sherbrooke-i egyetem informatikus hallgatója.

Ismét maratoni hétvége – Február 7–9.

Ahogy azt már két évvel ezelőtt is írtam, a linuxos társadalom számára az egyik legfontosabb feladat az, hogy a rendszer magyarítását, honosítását elvégezzük. Szerencsére nagy örömmel mondhatjuk, hogy a munka dandárját már elvégeztük, sőt nagyon sok rendszer rendelkezik magyar felülettel és leírással. A rendkívül sok emberórát igénybe vevő folyamat szerencsére a mai napig folytatódik és erősödik. Az egyik legtöbbet „szem előtt lévő” program, az OpenOffice.org irodacsomag magyarítása nagymértékben *Somogyi Péter* szervezőmunkájának köszönhető, aki több segítővel tavaly áprilisban maratoni „honosítási hétvégét” szervezett. Ma már mindenki könnyedén használhatja az elkészült rendszert, akár *.deb* csomagból is.

Sajnos a munkát tavaly nem tudtuk befejezni, a programban még pontatlan vagy részleges fordításokkal találkozhatunk, magyar súgó pedig még most sem tartozik hozzá. Hogy az eltelt idő alatt összegyűlt hibákat orvosolni tudjuk, Péter ismét egy maratoni hétvégét szervez. A tervek szerint húsz–huszonöt ember gyűlik majd össze egy erre megfelelő helyen, és a tavaly bevált rendszert is üzembe helyezve (amelyen keresztül külső segítők is bekapcsolódhatnak a munkába) a hétvége alatt az egész fordítást elkészítik. Ez első hallásra kivitelezhetetlennek tűnő feladat, a tavalyi tapasztalatok mégis azt mutatják, hogy a csapatnak ismét nagy esélye van megmutatni: a nyílt rendszerek élnek! Amennyiben kedvet és erőt érzel, hogy részt vegyél a hétvégén, vagy bármilyen módon támogatni tudod a csapatot (például ellátmánnyal, helyszín biztosításával), írj közvetlenül Péternek, vagy akár nekem! Várunk!

Kapcsolódó cím

OpenOffice.org magyarul ➲ http://office.fsf.hu

Szy György (Szy.Gyorgy@linuxvilag.hu)

Bevezetés a FreeS/WAN használatába (1. rész)

VPN-alagutak biztonságos vezeték nélküli és más világhálós kapcsolatok kiépítésére.

A legutóbbi, mintegy ötéves időszak során az IPSec a VPN-kapcsolatok vezető szabványává fejlődött. A FreeS/WAN (➲ http://www.freeswan.org) nyílt forrású, kiterjedt biztonságos számítógép-hálózat, a legnépszerűbb és az egyik legérettebb IPSec-megvalósítás, ami Linux-rendszereken működik. A mostani alkalommal és a következő hónapban azt fogjuk megvizsgálni, miért és hogyan kell a FreeS/WAN-t biztonságos hálózati adatcserére használni. Eme vizsgálódásunkat a vezeték nélküli (rádiós) hálózatokkal kezdjük.

VPN-alapok

Mindeddig a VPN-ek két leggyakoribb használati formája a két hálózat közötti kapcsolat volt. A telephelyek közti kapcsolatoknál mindegyik telephely, illetve hálózat saját VPN-átjáróval rendelkezik, azaz olyan VPN-kiszolgálógéppel, amely az IPSec-en vagy más VPN-protokoll alapú alagutakon keresztül tart fenn kapcsolatot más VPN-átjárókkal, ahogyan az az *1. ábrán* látható.

Ez a kiszolgálógép ugyanakkor a helyi hálózatra csatlakozó gépek számára útválasztóként is működik, hogy az egyes csomagokat más kapcsolódó VPN-helyekhez eljuttassa. Mindez más szóval azt jelenti, hogy egy telephelyek közti VPN-hálózatban több felhasználó, illetve számítógép osztozik egyetlen alagúton, hogy adatokat cseréljenek a távoli hálózatra csatlakozó számítógépekkel.

A távoli hozzáférésű VPN-ek, ideértve a vezeték nélküli hálózatokban használatos fajtát, ettől kicsit eltérőek. A VPN-alagút ahelyett, hogy egy teljes számítógépes hálózatot egy másikhoz kapcsolna, mindössze egyetlen felhasználó vagy számítógép és egy távoli hálózat között teremt kapcsolatot (*2. ábra*). A felhasználó helyi VPN-átjárója jellemzően alkalmazói program, amely a helyi gépen fut. A távoli VPN-átjáró rendszerint tűzfal vagy egy kijelölt VPN-készülék a gazdahálózaton.

A vezeték nélküli helyi hálózatból (LAN) épített VPN-ek a távoli hozzáférésű VPN-ek fontos alcsoportját alkotják. A vezeték nélküli hálózatok az általuk biztosított kényelemnek és az alacsony költségeknek köszönhetően egyre népszerűbbek. Minthogy azonban a nevüknek megfelelően az adatok közvetítéséhez rádióhullámokat használnak, könnyű őket lehallgatni.

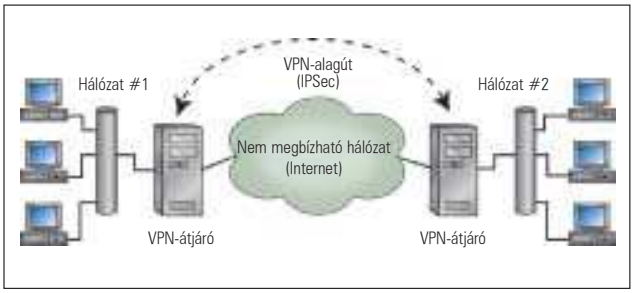
A hálózati termékek gyártói a vezetékes hálózatnak megfelelő titkosítás biztosítására tettek erőtlen kísérletet a vezeték nélküli titkosítási szabvány (Wireless Encryption Standard – WEP) megalkotásával, de a WEP-et a titkosítás megvalósításának hibái szinte azonnal elavulttá tették. Emiatt sok vezeték nélküli hálózatot használó vállalat máris elfordult a WEP-től. Helyette inkább VPN-alagutakat használnak a vezeték nélküli kapcsolatok titkosítására.

A 2. ábrához visszatérve meg kell említenünk, hogy egyetlen rendszer képes összetett VPN/vezeték nélküli átjáróként dolgozni. A *3. ábrán* újabb lehetséges hálózati elrendezést láthatunk: mind a vezeték nélküli egység, mind a VPN-átjáró önálló egységet alkot.

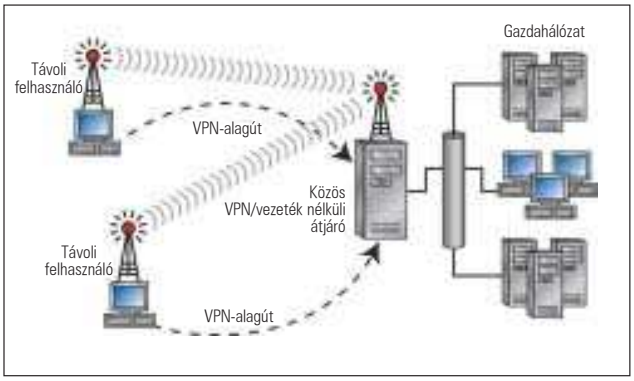
Az IPSec és a FreeS/WAN

Mint azt már korábban kijelentettem: az IPSec a legnépszerűbb VPN-protokoll. Minthogy ez az IP-protokoll kiterjesztése, ez az Internet „hivatalos” VPN-protokollja. Szinte az IPSec létezése óta ***John Gilmore*** és a FreeS/WAN kezdeményezés fejlesztőcsapata a FreeS/WAN csomag kifejlesztésével és a Linux számára történő átadásával minden tőle telhetőt megtett az IPSec minél szélesebb körű elfogadásáért.

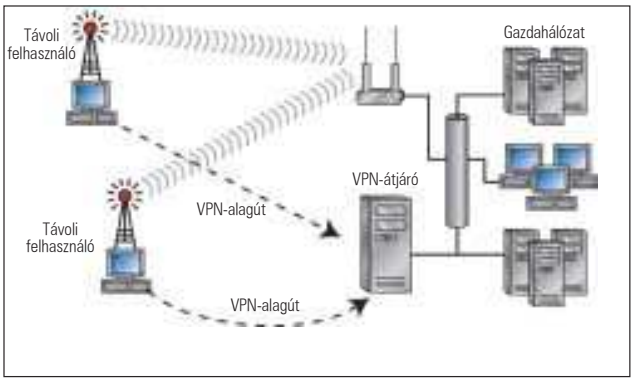
A FreeS/WAN legfrissebb változatáról szóló végleges adatok a kezdeményezés honlapjáról,



1. ábra A telephelyek közötti VPN-kapcsolat



2. ábra A vezeték nélküli – távoli hozzáférésű – VPN



3. ábra Egy másfajta LAN VPN-elrendezés

a ➲ http://www.freeswan.org címről szerezhetők be.
Egy szó, mint száz, a FreeS/WAN érett, jól dokumentált és támogatott program. Ha Linuxot használsz, akkor VPN-szükségleteid kielégítésére a FreeS/WAN a megfelelő választás.

A FreeS/WAN beszerzése és telepítése

A NetFilterhez hasonlóan a FreeS/WAN is a tényleges munkát végző rendszermagmodulból és a felhasználói felületekből áll, az utóbbiak a beállítást szolgálják. A NetFiltertől eltérően azonban a FreeS/WAN nem szerepel a Linux-rendszermag forráskódcsomagjai között, így a legtöbb változathoz adott rendszermagban nem is bukkan fel. Ez számos ország titkosítási eljárások korlátozása miatt van így.
A rendszermag újraépítése, esetleg újrafordítása a FreeS/WAN telepítése esetlen módjának tűnik. Ennek elkerülése érdekében számos Linux-változat, többek között a SuSE, a Debian és a Mandrake olyan FreeS/WAN-változatot tartalmaznak, amelyik képes az adott változathoz tartozó rendszermaggal együttműködni.
A Red Hat 7.3 Linux-változat felhasználói számára IPSec-képes rendszermagcsomagok – mind bináris, mind forráskód formában –, valamint a FreeS/WAN telepítőeszközei a Steamballoon telephelyéről, a ➲ http://rpms.steamballoon.com/freeswan címről tölthetők le.
Mivel jómagam leginkább SuSE és Red Hat Linux-változatokat használok, a FreeS/WAN program ezekhez a rendszerekhez történő beszerzését és telepítését fogom az alábbiakban ismertetni. Abban az esetben, ha az igényeid összetettebbek, leírás végett látogass el a ➲ http://www.freeswan.org/doc.html címre. Linux-változatodtól és a rendszermag frissességétől függően lehetséges, hogy a rendszermagot újra kell fordítanod, de ennek a folyamatnak a pontos leírása megtalálható a webhelyen.

A FreeS/WAN telepítése SuSE rendszerekre

Abban az esetben, ha a Linux-változathoz járó eredeti rendszermagot használod, egyszerűen telepítsd a sec-sorozatban szerepelő `freeswan.rpm` csomagot. Győződj meg róla, hogy az `ipsec.o` modul rendszermagváltozata megfelel az üzemelő SuSE rendszer rendszermagváltozatának.
Egy csapásra mindkét ellenőrzést elvégezhetjük: a futó rendszer magjának változatszámát a `uname -av` parancs kiadásával kérdezhetjük le. Az egyelőre telepítetlen `freeswan.rpm` csomag rendszermagváltozatát az alábbi paranccsal tudhatjuk meg:

```
# rpm -ql -p ./freeswan.rpm | grep -e ipsec.o
```

A rendszermag változatszámát az erre az állományra mutató útvonal fogja megmutatni, tehát a példának megfelelően */usr/lib/modules/2.2.18/ipv4/ipsec.o*. Ha a változatok azonosak, akkor telepítsd a csomagot:

```
# rpm -Uvh ./freeswan.rpm
```

Ezután a */etc/rc.config* állomány megnyitásával és a START_IPSEC változó értékének „yes”-re állításával kapcsold be az IPSec-et. Most érkezett el az ideje, hogy helyettesítsük a géphez tartozó titkosítási kulcsot – az RSA-aláírás kulcspárt –, amelyet a FreeS/WAN.RPM-ek telepítettek a rendszeredre.
Ha a */etc/ipsec.secrets* állomány létrehozási dátuma az aznapi (mai) dátumnál korábbi, akkor bizony új kulcsokra van szükség. Ennek megtételéhez a FreeS/WAN 1.92-es vagy annál magasabb változatszámmal ellátott változatoknál az alábbi parancsokat használhatjuk:

```
# mv /etc/ipsec.secrets /etc/ipsec.secrets.test
# ipsec newhostkey --hostname my.host.FQDN
➥--output /etc/ipsec.secrets --bits 2192
```

Természetesen a `my.host.FQDN` kapcsoló helyett a géped teljes nevét kell begépelned, például: `george.wiremonkeys.org`. A FreeS/WAN korábbi változatainál a következő parancsot használhatjuk:

```
# ipsec rsasigkey --hostname my.host.FQDN
➥2192  > /etc/ipsec.newkey
```

Ha használtad az `ipsec rsasigkey` parancsot, akkor szövegszerkesztővel meg kell nyitnod a */etc/ipsec.secrets* állományt, és az ott kapcsos zárójelek ({ }) között szereplő jelsorozatot ki kell cserélned az `ipsec.newkey`, vagy bármilyen más állományba irányított új kulccsal. Annak ellenére, hogy még be sem állítottad a szolgáltatást, máris elindíthatod és ellenőrizheted:

```
# /etc/init.d/ipsec start
# ipsec whack --status
```

Ha a második parancs (az `ipsec whack --status`) `000` visszatérési kóddal ér véget, FreeS/WAN-telepítésed megfelelően működik.

FreeS/WAN telepítése Red Hat-változatokra

Ha a Red Hat 7.3-as változatot a rendszerhez adott eredeti rendszermaggal üzemelteted – ez a cikk írásának idején a 2.4.18-as volt –, akkor először is szerezd be az IPSec-képes rendszermagcsomagot a
➲ http://rpms.steamballoon.com/freeswan címről, vagy a 44-es CD Magazin/FreeSWAN könytárból. Innen akár a forrást, akár a futtatóbinárist letöltheted; először a rendszermagcsomagot telepítsd.
Ha Linux-rendszeredben a rendszerhez adott eredeti rendszermagcsomag van telepítve – majdnem biztos, hogy az –, akkor használnod kell a `--force` kapcsolót, mivel a Steamballoon FreeS/WAN rendszermagcsomagjának bázisneve („kernel”) megegyezik a Red Hat-változat rendszermagjának csomagnevével. Celeron alapú Red Hat 7.3-as rendszeremen a Steamballoon rendszermagcsomagot így telepítettem (a példákban szereplő valamennyi változatszám mára már bizonyára elavulttá vált):

```
# rpm --force -i ./kernel2.4.18-3ipsec.i686.rpm
```

Felesleges aggódnod a `--force` kapcsoló kényszerítő jellege miatt, mivel a rendszermag lenyomatállományának nevei és a modulkönyvtárak mind egyediek, és régi rendszermagodat nem fogják felülírni. A saját Red Hat 7.3-as rendszeremet alapul véve az új rendszermag lenyomatállomány például a */boot/vmlinuz-2.4.18-3ipsec*, a modulkönyvtár pedig a */lib/modules/2.4.18-3ipsec* címen lesz elérhető. A Steamballoon RPM csomagjában levő telepítés utáni héjprogram felülírta a rendszermagra mutató bejegyzést a */boot/grub/grub.conf* fájlokba az új IPSec-rendszermagra mutató új bejegyzéssel – tulajdonképpen a betöltőmenüből eltávolította a régi rendszermagot. Ám amint a régi rendszermagra mutató bejegyzést ismét hozzáadtam a menühöz, rendszerbetöltéskor gond nélkül tudtam választani közülük.
A rendszermagcsomag telepítését követően telepítsd a felhasználói módú eszközöket. Az én rendszeremen ez az alábbi

paranccsal történt meg:

```
# rpm -i freeswan-1.97-0.i386.rpm
```

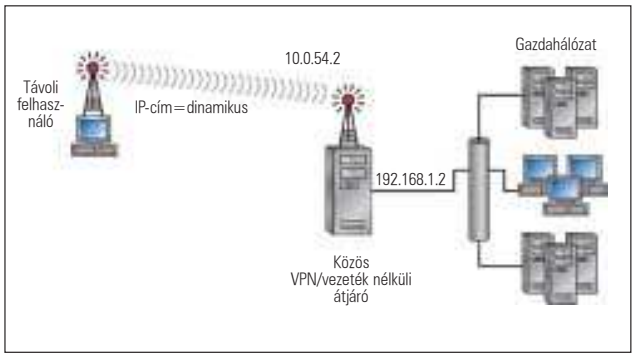
Az RPM csomag egy héjprogramot – */etc/init.d/ipsec* – telepít, de a szolgáltatást nem indítja el. Az üzembe helyezést az alábbi paranccsal végezhetjük el:

```
# chkconfig --add ipsec
```

Ezután hozz létre új RSA-kulcspárt, úgy, ahogy azt az előző szakaszban megadtam. Ha kész, máris elindíthatod és kipróbálhatod a FreeS/WAN telepítésedet szolgáltatást a már ismertetett módon.

Vezeték nélküli LAN VPN beállítása

Sajnos helyszűke miatt csak egyetlen gyakori alagutazás forgatókönyvét tudom megvizsgálni – ezt mutatja be a 2. ábra. A *4. ábra* a 2. ábrán látható hálózat egy részletét tárja fel, ezúttal



4. ábra IP-címekkel rendelkező vezeték nélküli VPN

IP-címekkel ellátva. Mint azt rövidesen látni fogjuk, nincs szükség az összes lehetséges ügyfél IP-címének ismeretére. A cikk hátralevő része több fontos alapfeltevésen nyugszik:

1. Az alapvető hálózati kapcsolatok működnek, a vezeték nélküli ügyfelek képesek kapcsolódni a kiszolgálóhoz.
2. Az alapvető csomagtovábbítás működik: a vezeték nélküli ügyfelek képesek elérni az átjáró túloldalán lévő gépeket.
3. Az átjáró egyelőre még nem működik tűzfalként.

A harmadik feltevés csupán könnyítés. A biztonság rétegekben valósítható meg, és a telepítési időn kívül semmibe sem kerül, hogy megszabjuk, a bejövő VPN-forgalom merre mehet és merre nem. Sajnos itt nem bocsátkozhatom mélyebb részletekbe; ehelyett olvasd el a ***FreeS/WAN quick start on firewalling*** HOGYAN-t a ➲ http://www.freeswan.org/doc.html címen.

Az ipsec.conf elkészítése

A FreeS/WAN beállítása két állományon keresztül történik: a */etc/ipsec.conf* a főbb beállításokat tartalmazó állomány, míg a */etc/ipsec.secrets* a kulcstár. Mindkét állománynak korlátozott engedélyekkel kell rendelkeznie; a 0600 megbízható választás. A */etc/ipsec.secrets* állományt különös gonddal kell védelmezni. Amennyiben ebből az állományból bármit ki kell másolni, mondjuk a gép nyilvános RSA-kulcsát, a szükséges adatot másold ki egy önálló állományba. Soha ne engedd meg, hogy az ***ipsec.secrets*** állomány kikerüljön a rendszeredből. Ezt az állományt

1. lista /etc/ipsec.conf a vezeték nélküli ügyfélen

```
# alapbeÆll tÆs
config setup
     interfaces=%defaultroute
     klipsdebug=none
     plutodebug=none
     plutoload=%search
     plutostart=%search
     uniqueids=yes

conn %default
     keyingtries=0
     authby=secret

conn george-gracie
     authby=rsasig
     left=10.0.54.2
     leftid=@george.wiremonkeys.org
     leftsubnet=0.0.0.0/0
     leftrsasigkey=0sAQPF0JJvY7...
     right=%defaultroute
     rightid=@gracie.wiremonkeys.org
     rightrsasigkey=0sNU0q2Y0Y0jxAIKuutV3...
     auto=start
```

részletesebben a következő hónapban fogjuk megvizsgálni. Az *1. lista* egy vezeték nélküli VPN-ügyfél ***ipsec.conf*** állományát mutatja be. A */etc/ipsec.conf* állomány három részre tagolódik: az alapvető telepítési kapcsolókra (`config setup`), az előre beállított alagútértékekre (`conn %default`), és végül az alagút-meghatározásokra (`conn george-gracie` az 1. listában, ezt a nevet magam választottam ennek az alagútnak). A `config setup` szakaszban található előre beállított jellemzők az egykártyás rendszereken biztonsággal érintetlenül hagyhatók. A legfontosabb beállítás az `interfaces`, ami megadja, hogy melyik hálózati csatoló lesz az IPSec-csatornák helyi végpontja. Az előre beállított érték, a `%defaultroute` ennek az `ipsec=[csatol ]` a feloldása, ahol a `[csatol ]` a gép alapértelmezett hálózati csatolója, általában az `eth`. Ezennel máris az ***ipsec.conf*** lényegéhez érkeztünk el: az alagút-meghatározáshoz. Az 1. listában a george-gracie alagút-meghatározásban az első sorban találjuk az `authby` beállítást, ami azt szabja meg, hogyan végezzék az egyes IPSec-gépek a saját hitelesítésüket egymás számára. Az alapérték a „secret”, amely az előzetesen megosztott titkos azonosítót jelenti. Ez a beállítás lehetővé teszi, hogy az előre meghatározott titkos karaktersorozat legyen az azonosítási kulcs. Ez elsőre meggondolatlanságnak tűnhet, de mégis biztonságos, ugyanis az azonosító sosem utazik a hálózaton.
Most más a helyzet, hiszen az 1. listában az `authby` kapcsoló `rsasig`-re van beállítva. Az RSA-hitelesítés nem szükségszerűen sokkal biztonságosabb, ezzel szemben sokkal kényelmesebb. Míg a megosztott azonosítót valamilyen biztonságos eszköz, mint például titkosított levél vagy az SSH segítségével előzetesen ki kell cserélni, az RSA-hitelesítésben használt nyilvános kulcsok akár ország-világ színe előtt kicserélhetők, mi több, még a honlapon is közzétehetők.
A FreeS/WAN beállításánál fontos a bal és a jobb oldal megkülönböztetése: ezek a parancsokban és kapcsolókban az IP-alagutak végpontjainak megjelölésére szolgálnak. Hogy melyik

2. Lista /etc/ipsec.conf az átjárón

```
config setup
     interfaces="ipsec0=eth1"
     # Ös  gy tovÆbb
     forwardcontrol=yes

conn george-gracie
     authby=rsasig
     left=10.0.54.2
     leftid=@george.wiremonkeys.org
     leftsubnet=0.0.0.0/0
     leftrsasigkey=0s0sAQPF0JJvY7xK9Cmx1...
     right=%any
     rightid=@gracie.wiremonkeys.org
     rightrsasigkey=0sNU0q2Y0Y0jxAIKuutV3...
     auto=add
```

elnevezés melyik oldalt jelöli, tulajdonképpen lényegtelen, az a fontos, hogy a jelöléseket következetesen alkalmazzuk. Ugyanannak a gépnek mindkét alagút-meghatározásban jobb oldalinak kell lennie, és egy rendszernek sem szabad megfordítania a másik gépen szereplő bal–jobb kiosztást.

A biztonság kedvéért, amikor egy gép egy másik számítógép számára kiszolgálóként működik, ezt a gépet javasolt bal oldaliként feltüntetni. Példánknál maradva a ***George*** nevű gép kiszolgálóként működik a vezeték nélküli ügyfelek számára, így ***George*** bal oldali, míg ***Gracie***, az ügyfélrendszer, jobb oldali.

A „bal" alagútbeállítási értéke jelzi az alagút bal oldali végpontjának IP-címét, ***George***-ot: 10.0.54.2. A jobb oldal természetesen a jobb oldali végpont IP-címét jelöli ki. Forgatókönyvünkben viszont az áll, hogy ***Gracie*** és a többi vezeték nélküli ügyfél dinamikus IP-címet kap. Ekkor egy rögzített IP-cím megadása helyett a `%defaultroute` értéket fogjuk használni, amely a gép alapértelmezett útvonalát jelöli; ez a hálózati kártyához tartozó IP-címmé fog alakulni.

Az alagút `leftid` kapcsolója felesleges ismétlődésnek tűnhet, minthogy a `left`-tel már azonosítottuk a bal oldali IP-címet, ám ez valójában egy kicsit eltér attól. A `leftid` és `rightid` kapcsoló minden alagút végponthitelesítési azonosítóját határozza meg. Ez lehet egy IP-cím, de lehet a kukac (@) jelet követő teljes név (FQDN). Minthogy a ***Gracie*** ügyfél, a dinamikus IP-cím kiosztáson keresztül kap címet, így a `leftid` számára a @gracie.wiremonkeys.org az egyetlen használható érték. Ebben a példában ugyanakkor a `rightid` számára a @george.wiremonkeys.org és a 10.0.54.2 IP-cím kölcsönösen felcserélhető értékek.

Az 1. listában a következő megfontolásra érdemes alagútbeállítás a `leftsubnet`. Ez meghatározza, hogy milyen cél-IP-címek fogadhatnak csomagokat a jobb oldaltól, emiatt azt is, hogy a jobb oldali végpontok milyen célokhoz használhatják az alagutat. ***Gracie*** és a többi vezeték nélküli ügyfél ***George***-ot használja egyedüli átjáróként a vállalati helyi hálózathoz és általában a nagyvilággal való kapcsolattartásra, ezért ezt 0.0.0.0/0-ra állítottuk, ami így az összes címet kifejezi.

Létezik egy `rightsubnet` kapcsoló is, ezt azonban kizárólag a telephelyek közti kapcsolatokban szükséges beállítani. Távoli hozzáférés vagy egyéb kiszolgálóalapú megoldásokban a két kapcsoló közül csak az egyik beállítása szükséges.

Ha az alagút számára RSA-hitelesítést írtunk elő, akkor mind a `leftrsasigkey`, mind a `rightrsasigkey` megadása kötelező. Ezek az értékek az ***ipsec.secrets*** állományban, a `#pubkey=` kezdetű sorokban találhatók meg. Az alábbi két, vagylagos jelleggel használható parancs közül választhatunk:

```
# ipsec showhostkey --left
```

vagy

```
# ipsec showhostkey --right
```

A `--left` és `--right` kapcsolók (amelyek a FreeS/WAN 1.9-es vagy annál későbbi változatában használhatók) révén a kimenet a `leftrsasigkey` vagy `rsasigkey` utasítás formátumának megfelelő alakú lesz, ami aztán az ***ipsec.conf*** állományba rugalmasan bemásolható lesz. Például a ***George*** gépen futtatott `ipsec showhostkey --left` az 1. listában bemutatott `leftrsasigkey` értéket, a ***Gracie*** gépen futtatott `ipsec showhostkey --right` pedig a `rightrsasigkey` értéket adta eredményül. Fontos megemlíteni, hogy bár az RSA-kulcsok hosszúak, mindegyiküknek el kell férnie egyetlen sorban, vagyis sortörés nem használható!

Az utolsó kapcsoló az 1. listában az alagútbeállításánál az `auto`, ami – amint az IPSec elindul – meghatározza a FreeS/WAN számára, hogy van-e tennivaló az alagút körül. A `start` érték írja elő, hogy a kezdeti értékadás és az önműködő indítás megtörténjen. Az `add` érték előírja, hogy a `pluto` nevű IPSec-démonhoz legyen hozzáadva. Az `ignore` hatására pedig figyelmen kívül hagyja az alagutat. Az 1. listában az `auto` értéke `start`-ra lett beállítva, ezért amikor csak az IPSec a ***Gracie*** gépen elindul, szeretnénk felélesztení az alagutat a ***George***-on. Az `auto` számára a `plutoload` és `plutostart` telepítésbeállítási kapcsolókat megfelelően kell beállítani azaz, hogy valamiféle értelmük legyen: részletesebb útmutatáshoz az ***ipsec.conf(5)*** súgóoldalról juthatunk.

Rendben, ez minden, amit az ***ipsec.conf*** ügyféloldali beállításáért tenni kell. De vajon mi a helyzet a kiszolgálóval? Úgy fest a dolog, hogy ebben a forgatókönyvben a beállítás mindkét oldalon közel azonos. A 2. lista mutatja a ***George*** nevű gép */etc/ipsec.conf* állományát.

Az első eltérés a ***Gracie***-géptől az, hogy ***George*** több hálózati kártyával rendelkezik, következésképpen a hálózati csatolót egyértelműen meg kell határozni. A kiszolgáló alapértelmezett útvonala a helyi hálózatra csatlakozik, és nem a vezeték nélküli hálózatra, ennek következtében a `%defaultroute` értéket nem használhatjuk.

Ezen kívül egy új telepítési kapcsolóval is rendelkezünk, a `forwardcontrol`-lal. Ha ennek értékét „yes”-re állítjuk, elő fogja írni az IPSec-nek, hogy szükség szerint kapcsolja be az IP-továbbítást, és kapcsolja ki, amint az IPSec lezárásra kerül. Továbbá magában az alagútszakaszban a `right` (jobb oldal) kapcsolót `%any`-re kell állítani, nem peidg `%defaultroute`-ra, hiszen a `%defaultroute` *George* helyi hálózatát adná vissza, és nem *Gracie*-t, vagyis a jobb oldalt. Ezenkívül az `auto` kapcsolót `start` helyett `add`-re kell állítani, mert *George* kiszolgálóként működik: csak készen kell állnia *Gracie* számára, hogy az alagutat működésbe helyezze.

Az alagút elindítása és kipróbálása

Most ugrik a majom a vízbe! Előbb *George*-on, majd *Gracie*-n adjuk ki a következő parancsot:

```
ipsec setup restart
```

George végigolvassa a */etc/ipsec.conf* állományt, betölti a `george-gracie` alagút-meghatározást, és felkészül a kapcsolatok fogadására. *Gracie* ugyanezt fogja tenni, és felélesztí az alagutat. Az indulással kapcsolatos üzenetek naplózása a */var/log/messages* vagy a */var/log/secure* állományokba történik. Amennyiben az ügyfélrendszeren `ipsec setup restart` parancs kimenete az „IPSec SA established” üzenettel ér véget, az alagút működésre kész!
Próbáld pingelni a távoli hálózat gépeit, vagy kísérelj meg valamilyen más módon kapcsolódni hozzájuk – a kapcsolatnak nem szabad különböznie az alagút üzembe helyezése előtt tapasztalt működéstől!

A `tcpdump` futtatásával hasznos lehet meggyőződni arról, hogy a kártyán kizárólag ESP- (Encapsulating Security Payload) csomagok közlekednek, és nem tényleges ping, FTP-, vagy egyéb csomagok vannak kiküldve.
A következő hónapban egy-két újabb forgatókönyvet fogunk áttekinteni, és még jobban elmélyedünk a FreeS/WAN gyönyörűséges rejtelmeiben.
Remélem, írásom elegendő útbaigazítást nyújtott a biztonságos vezeték nélküli hálózathasználat megkezdéséhez.

Linux Journal 2003. január, 105. 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”.

KAPCSOLÓDÓ CÍMEK

A FreeS/WAN honlapja ➲ http://www.freeswan.org/
FreeS/WAN-levelezőlista ➲ users@lists.freeswan.org
FreeS/WAN-cikk a SysAdmin-on
➲ http://www.samag.com/documents/s=1159/sam0011i/0011i.htm

Karakteres munkamenet-kezelő

Adam a Screen sokféle előnyét és használati módját mutatja be.

A Screen olyan terminálnyaláboló (terminal multiplexer) program, amely sok folyamat kezelését teszi lehetővé egyetlen terminálon keresztül. Minden egyes folyamat saját virtuális ablakot kap, és a folyamatokkal kölcsönhatásban lévő virtuális ablakokat szabadon váltogathatjuk. A Screen által kezelt folyamatok akkor is tovább futnak, ha az ablakuk éppen nem működik.
Mindeddig tehát a Screen által nyújtott szolgáltatások nem túl érdekesek, de még csak nem is újak. Voltaképpen már léteznek olyan X11 terminálalkalmazások, amelyek képesek biztosítani ezt a szolgáltatást: a `konsole` és a `multi-gnome-terminal`. A Screent a többi programtól néhány kulcsszolgáltatás különbözteti meg.
A Screen képes leválni egy munkamenetről (session), és ahhoz egy későbbi időpontban újra csatlakozni. A munkamenetről történő leválasztás után a folyamatokat figyelő Screen továbbra is fut. Amennyiben a későbbiek folyamán újra csatlakozol a munkamenethez, termináljaid ugyanúgy várnak rád, ahogyan otthagytad őket.
A Screen minden egyes általa kezelt ablak számára egyedi, visszakereshető és lapozható vagy görgethető (scrollback) tárterületeket (puffereket) tart fenn. Megadhatunk „add meg a kívánt szót, és én megkeresem neked” típusú keresési parancsokat, de növekményes keresési műveleteket is végezhetünk. Ez annyira magától értetődő szolgáltatás, hogy már szinte az a meglepő, hogy több terminálemulátor nem ajánlja fel.
A Screen említésre méltó szolgáltatásai között megtalálhatók a beállítható billentyű-hozzárendelések, az `utf8` és többféle karakterkészlet-támogatás, többszörös csatlakozás (multi-attach), beállítható bementi és kimeneti fordítás, bemeneti és kimeneti szűrő hozzáférés-szabályozó listákkal (Access Control List – ACL), több felhasználó támogatása, valamint naplózás.

Kapcsolatban a Screennel

Mielőtt ténylegesen futtatnánk a Screent, fontos megértenünk, miként kell vele együttműködnünk. A Screen a vezérlőkarakter kivételével az összes bevitt szöveget elküldi a pillanatnyilag működő ablaknak.
Az előre beállított vezérlőkarakter a CTRL-A – tartsuk lenyomva a CTRL billentyűt, miközben leütjük az A billentyűt. A Screen súgóoldal, az Emacshoz hasonló módon, a CTRL jelölésére a C billentyűt használja.
A vezérlőkarakter arra használható, hogy tudassuk a Screennel, ezúttal magának a Screen programnak a működését szeretnénk szabályozni, nem pedig az adott ablakban megjelenő alkalmazását.
A vezérlőkarakter után lenyomott billentyű jelzi, hogy melyik Screen parancsot szeretnénk végrehajtani. Néhány lényeges parancsot és az azokhoz tartozó billentyű-hozzárendeléseket *táblázatunkban* foglaltuk össze. Sok gyakran használt parancsnál a billentyűműködés-szabályozó változata is parancshoz van kötve. Példaként említhetjük az ablak létrehozásához használatos CTRL-A C és a CTRL-A CTRL-C billentyűket. Ahhoz viszont, hogy a CTRL-A kombinációt egy alkalmazáshoz eljuttathassuk, a CTRL-A A kombinációt kell használnunk.
Ha szükséges, a vezérlőkarakter más karakterre cserélhető. Az Emacs-felhasználók a vezérlőkaraktert jellemzően CTRL-B-re szokták cserélni oly módon, hogy a ***.screenrc*** állományukat az `escape Bb` sorral egészítik ki. Valamennyi alábbi példánkban viszont a CTRL-A billentyűegyüttest használjuk, mert ez az alapértelmezés.
Amint az várható, a ***.screenrc*** az adott felhasználóhoz tartozó saját könyvtárban (***/home***) a felhasználókra külön-külön jellemző saját beállításokat tartalmazza, miközben a ***/etc/screenrc*** állomány az egész rendszer működését érintő beállítóállomány, amely az összes felhasználóra érvényes.

Ablakkezelési alapok

Most, hogy már tisztáztuk a Screennel való „együttműködés” alapjait, végigfuthatunk a Screen-alapismereteken, példaként véve egy jellemző Screen-munkamenetet. A Screennek valószínűleg legjellemzőbb használata az olyan távoli géphez csatlakozó terminálok irányítása, amelyekre felhasználói azonosítóval rendelkezünk.
Azok, akik eddig csak játékkal töltötték otthon az időt, most jelentkezzenek be egy távoli gépre, amelyre már telepítették a Screent. Ha nem tudunk olyan gépről, ahová `ssh`-val be lehetne lépni, akkor telepítsd a Screent és `ssh`-val lépj be a helyi gépre (localhost).
A Screen-csomagok a legtöbb Linux-változat számára elérhetők. Most tehát előttünk van a távoli gép parancssora.
A parancssorba gépeljük be a `screen` szót. Ekkor a bejelentkező-képernyőnek kell megjelennie, amelyen tájékoztató adatok láthatók arról, hogy a Screen a GPL szabályozás alá tartozik, valamint azt, hogy hol lehet jelezni az esetlegesen felfedezett hibákat és más ehhez hasonló adatokat. A bemutatkozó képernyőt a szóközbillentyű lenyomásával hagyhatjuk el, de a ***.screenrc*** állományban a `startup_message` változó értékének `off`-ra állításával végérvényesen ki is kapcsolhatjuk. Ezután újabb parancssorhoz kell érkeznünk, ámde ez már a Screen programon belül fut.
A Screen programon belüli héjprogramnak ugyanúgy kell működnie, ahogyan az első tenné. Ha kiadod a `printenv` parancsot, akkor láthatod, hogy néhány új környezeti változó is be lett állítva. A Screen program a `TERM` változó értékét `screen`-re állítja, minden egyes ablak vt100-megfelelő virtuális terminált biztosít. A `WINDOW` változó értéke a virtuális ablakszámra, az `STY` változó pedig a munkamenet (session) nevére lesz állítva. Az utóbbi két fogalmat a későbbiekben fogom elmagyarázni.
Mindeddig ez az egyszerű rendszerhéj (shell) pontosan úgy működik, ahogyan azt egy távoli gépen már megszokhattad. A példa kedvéért kezdjük a Screen legfrissebb változatának – ez az írás létrejöttekor a 3.9.13 volt – letöltésével a Screen program terjesztési helyéről, a ➲ ftp.uni-erlangen.de/pub/utilities/screen címről. Az állomány letöltődése közben, az üres idő hasznosítása végett, dönthetsz a saját könyvtárad

A parancsok és a hozzájuk kapcsolódó billentyűkombinációk

A parancs neve	A billentyű	Az általa végzett művelet leírása
screen	C	Új ablak létrehozása.
kill	K	Ablak bezárása.
help	?	A választható parancsok és billentyűkombinációk listázása.
detach	D	Leválasztás a screen-munkamenetről.
next	N	A következő ablak kijelölése.
prev	P	Az előző ablak kijelölése.
other	CTRL-A	Átváltás az előző ablakra.
windowlist	"	Az összes screen-ablak megjelenítése. Ez a billentyű lehetővé teszi a lista egyik elemének a kiválasztását.
copy	[	Átkapcsolás, másolás és beillesztés, illetve visszalapozás üzemmódba.
break	B	Megszakítás (break) küldése.
lockscreen	X	A pillanatnyi munkamenet bezárása.
kettős pont	:	A screen-parancssor elérése. (A kettős pont használata hasonló a vi szerkesztőben megszokotthoz).
Select	'	Megadhatjuk a kívánt munkamenet számát vagy nevét, amire váltani szeretnénk.
windowlist -b	"	Kilistázza az összes munkamenetet, a nyílbillentyűkkel választhatunk közülük.
select 0	0-9	0 től 9-ig vált a munkamenetek között.
title	A	A munkamenet nevét adhatjuk meg.
clear	C	Kipucolja a képernyőt.

(*/home*) kitakarítása mellett. Ha nem használnánk a Screent, akkor egy újabb `xterm`-ablakot kellene nyitnunk, és `ssh`-val be kell lépnünk a távoli gépre. A Screen programban viszont a CTRL-A, C billentyűk új ablakot hoznak létre, új héjfolyamattal.
Eddig tehát rendelkezünk egy FTP-ügyféllel meg egy héjprogrammal, amelyik szorgalmasan tisztogatja a saját könyvtárat. Közben a CTRL-A, P billentyűkkel az eredeti FTP-ablakban ellenőrizhetjük a letöltést – ezzel a paranccsal az előző ablakba jutunk vissza.
A héjprogramhoz a CTRL-A, N billentyűkkel térhetünk vissza. Ellenőrizzük a letöltést, hogy befejeződött-e már. A Screen letöltése egyébként nem vesz olyan sok időt igénybe, de a példa kedvéért most úgy teszünk, mintha mégis sokáig tartana. Ugye, most már ideje lenne visszatérni a kacatokkal telezsúfolt saját könyvtárhoz? Mielőtt ezt megtennénk, nyomjuk le a CTRL-A, M billentyűket, hogy figyelemmel kísérhessük az éppen működő ablak kimenetét. A Screen most értesítést fog küldeni, ha az FTP-ablakban valamilyen tevékenység zajlik. A letöltési folyamat ellenőrzéséhez ezentúl már nincs szükség az ablakok váltogatására. Sőt ez fordított esetben is működik: használjuk a CTRL-A, _ billentyűket, hogy a tevékenység nélküli időszakot figyelemmel kísérhessük, ez alapértelmezés szerint 30 másodperc. A tevékenység nélküli időszak megfigyelése különösen hasznos hosszú programfordításoknál vagy más olyan munkáknál, amelyek ontják az adatokat.
Sorozatban további héjprogramokat tartalmazó ablakokat hozhatunk létre, és ezzel párhuzamosan végezhetjük tennivalóinkat a távoli gépen. Néhány ablak létrehozása után nehéz nyomon követni, hogy melyik ablak hol is található. Ezen a ponton jut szerephez a `windowlist`. Nyomjuk meg a CTRL-A," billentyűket, mire megjelenik az éppen megnyitott ablakokat tartalmazó lista. A listában a J és a K billentyűkkel lehet mozogni. Amikor a lista kijelölt bejegyzésén állva leütjük az ENTER billentyűt, az válik a pillanatnyi ablakká. Az előre beállított értéknek megfelelően az ablaknév nem valami beszédes. Ezen a helyzeten úgy lehet javítani, hogy az ablak nevét a CTRL-A, SHIFT-A billentyűkombinációkkal magunk állítjuk be. Ezeket a címeket önműködően is be lehet állítani, akárcsak az `xterm`-ben tennénk: egy ESC-K, majd a cím, végül az ESC-\ jelsorozat elküldésével.
Nagyon valószínű, hogy magad is képes leszel alkalmazni a héjprogramnak jobban megfelelő receptet, hogy a megadott átváltási sorozattal (escape sequence) az `xterm` címsorát a `screen` ablak nevére változtasd.
Az ablakkezeléssel rohamléptekben zajló ismerkedésünk befejezéseként zárjuk be az összes ablakot. Ha kilépsz a Screen által gyártott héjprogramból, egyúttal az ablak is önműködően törlődik. Az ablak a `kill` parancs segítségével kézzel is törölhető, az alapértelmezés a CTRL-A, K. Ha az összes Screen ablakból kilépünk, a Screen is befejezi működését. A Screen a `quit` paranccsal arra is utasítható, hogy zárja be az összes ablakot, és lépjen ki – ezt a CTRL-A, \ billentyűkkel lehet gyorsan megtenni.

Munkamenetek

Ha egymás után hozol létre új ablakokat és váltogatod őket, lehetőséged nyílik rá, hogy teendőidet párhuzamosan végezd. Használhatsz mondjuk egy-két szövegszerkesztőt, IRC-csevegőt és még jó néhány további programot, mindegyiket a saját ablakában. Az esetenként előforduló katasztrófa azonban mindig váratlanul történik, és a hálózati kapcsolat megszakad. Akik még mindig játszottak az otthoni gépükön, most aztán bezárhatják SSH ügyfélprogramjukat. Ugye, úgy tűnik, ideje elkezdeni összeszedni a rendszer darabjait, és újra elindítani az alkalmazásokat a távoli gépen? Nos, ha Screent használunk, akkor erről szó sincs.
Minden alkalommal, amikor a Screent kapcsolók nélkül indítjuk el, az új munkamenetet hoz létre. Ez két folyamatot kelt életre: egy terminálkezelő folyamatot és egy ügyfélfolyamatot. Az ügyfélfolyamat önműködően hozzá van kapcsolva a terminálkezelési folyamathoz. Amikor tehát gépelünk, a begépelt karakterek az ügyfélprogramhoz kerülnek, ami a megfelelő alkalmazáshoz küldi őket tovább.
Abban az esetben, ha a hálózati kapcsolatod megszakad, az ügyfélprogram veszi a jelet, és leválik a terminálkezelési folyamatról. A terminálkezelési folyamat tovább dolgozik, figyelemmel kísérve a terminálokat, mintha mi sem történt volna. Amikor újra visszajelentkezel a rendszerbe, a parancssorban kiadott `screen -ls` paranccsal listát készíthetünk a futó munkamenetekről. Ennek az alábbi üzenethez hasonlót kell eredményeznie:

```
There are screens on:
          24319.pts-9.hostname (Detached)
1 Sockets in /var/run/screen/S-userid.
```

Gnome-Multi-Terminal

A cikkben a szerző is említést tesz a Gnome grafikus munkafolyamat-kezelőről, ami tulajdonképpen ugyanazt hivatott ellátni, mint a `screen`, csak grafikus elemekkel van bővítve. Ugyanúgy új terminálablakokat nyithatunk, amelyeket itt különféle fülek jeleznek, és a közöttük történő váltást is a segítségükkel tehetjük meg. A beállítás menüben a felületet testreszabhatjuk, vagyis hogy mi jelenjen meg és mi ne az ablak keretén belül. Választhatunk a *Toolbar* vagy a *Buttonbar* között; mindkettőt nem érdemes használni, mivel ugyanazokat a lehetőségeket kínálják. A *Toolbar* nagyobb helyigényű, mint a *Buttonbar*, így egyértelműen az utóbbi mellett szavazok. Az én `screen`-emnél tapasztalt véletlenszerű lefagyások miatt kénytelen voltam átszokni a `gnome-multi-terminal`-ra és a grafikus felületre, ami most már nem zavar annyira, mint az elején. Sokszor nyomkodtam a megszokott billentyűket, megszokásból, sajnos ezeket itt nem tudtam használni. Ennél is lecserélhetjük az egyes folyamatokhoz tartozó neveket valami sokatmondóbbra, tehát nem kell tíz *Shell* nevű ablak között habozni, hanem különféle pontos meghatározásokat írhatunk hozzájuk (például `tail -f syslog`).

Csontos Gyula (Csontos.Gyula@linuxvilag.hu)

Ez az üzenet azt mutatja, hogy a munkamenet önműködően vált le, amikor a kapcsolat lebontott.
A munkamenethez való újrakapcsolódásra többféle lehetőség is kínálkozik. Egyértelműen meghatározhatjuk a munkamenet nevét:

```
# screen -r munkamenetnev
```

A munkamenet számára előírhatod, hogy csatlakozzon újra, ha ez lehetséges, különben pedig kezdjen új munkamenetet:

```
# screen -R
```

Választhatjuk még a „tedd, amit kell, csak megkapjam a Screen-gépemet” elméletet, és használjuk a `screen -D -RR` parancsot. Eme legutóbbi lehetőség a már csatlakozott ügyfeleket is leválasztja, és a felsorolásban szereplő első munkamenethez rendeli hozzá.
Amikor ezen parancsok valamelyikét használod, tudnod kell, hol hagytad abba a munkát, még mielőtt még hálózati kapcsolat tönkrement volna. Az újracsatlakozás után a munka úgy folytatható, mintha mi sem történt volna.
Egy munkamenethez való többszöri csatlakozásra is adott a lehetőség. Ennek akkor vesszük hasznát, ha a Screen-munkamenetet egy másik gépről még nem zártuk le, vagy csak egyszerűen ugyanazonabból a munkamenetből szeretnénk ablakokat egymás mellé sorakoztatni. A többszörös csatlakozáshoz a parancssorban a `screen` parancs kiadásakor használjuk a `-x` kapcsolót.
Végül, amikor közeleg a munkanap vége, és elérkezik a hazaindulás ideje, a munkamenetről való leválás a CTRL-A, D parancs használatával érhető el. Másnap, amikor újra csatlakozol, ugyanott leszel, ahol a munkát abbahagytad.

A másolás és beillesztés, illetve a visszalapozás üzemmód

A Screennek írásunk elején említett egyik kulcsfontosságú szolgáltatása a visszakereshető visszalapozás (searchable scrollback). Ez egy olyan szolgáltatás, amelyet sohasem tudtam nélkülözni. Az újdonsült Screen-felhasználók számára ez nem értetődik azonnal magától, de a Screen visszalapozása a másolás parancson keresztül érhető el. Másolás módba a CTRL-A, [billentyűkkel vagy a `copy` paranccsal lehet kerülni. A navigáció a várakozásoknak megfelelően működik: használhatók a kurzormozgató nyilak, a PageUp PageDown lapozóbillentyűk, vagy a `vi` szerkesztőprogramban megszokott megfelelőik.
A `vi`-stílusú kereséshez a / (perjel) vagy ? (kérdőjelek) használandók, növekményes kereséshez a CTRL-S, illetve a CTRL-R billentyűkombinációk. A kis- és nagybetűk megkülönböztetése az `ignorecase yes` paranccsal kapcsolható ki.
Amennyiben a másolásmódot kizárólag a visszalapozáshoz használjuk, akkor ebből az ESC billentyű lenyomásával bármikor kiléphetünk.
Szöveg másolásához először igazítsuk a kurzort (a kurzort helyőrnek is hívják – a szerk.) a másolandó szöveg elejéhez, majd a kijelöléshez üssük le a szóközbillentyűt. Ezután vigyük a kurzort a másolni kívánt szöveg végéhez, és a kijelölés befejezéséhez ismét nyomjuk le a szóközbillentyűt. Amint a másolandó szöveg végét kijelöljük, a szöveg a másoláshoz használt belső ideiglenes tárterületbe kerül, egyúttal a másolási üzemmódból is kilépünk. Utána ebből az átmeneti tárterületből a szöveget a CTRL-A] billentyűk segítségével a működő ablakba másolhatjuk.
Az utolsó jellemző, amelyet a másolás–beillesztés módról tudnunk kell, az a visszalapozási átmeneti tár, ami az előre beállított érték szerint száz sorra terjed ki. Véleményem szerint ez nem elegendő. Az értéket magasabbra állíthatjuk, ha a ***.screenrc*** állományba beszúrjuk a `defscrollback 1024` parancsot tartalmazó sort.

Ablakok megnyitása a .screenrc-vel

Mint már fentebb említettem, a ***.screenrc*** állományt parancsokkal egészíthetjük ki, hogy Screen viselkedését a saját igényeinkhez igazítsuk. Nem magától értetődő, hogy a `screenrc`-be bármilyen Screen-parancs beírható. Ez roppant hasznos, és az ablakgyártást (spawn) rögtön a `screen` parancs kiadásakor önműködően meg lehet kezdeni.
E csemegének számító ismeret alkalmazása a Screennek előre meghatározott ablakkészlettel való indítása. A következőkben egy éppen ilyen feladatokat ellátó minta ***.screenrc*** állományt láthatunk:

```
# elısz r olvassuk be a sajÆt screenrc
# ÆllomÆnyunkat mielıtt bÆrmi mÆst tesz nk
source $HOME/.screenrc

# most pedig kezdj k el az ablakok kinyitÆsÆt
screen top

# az ablak c msorÆban lØvı c met
# megvÆltoztathatjuk a -t lehetısØg rØvØn
# megvÆltoztathatjuk
screen -t irc epic

# meghatÆrozhatjuk az ablak szÆmÆt,
# melyikben induljon az alkalmazÆs
screen -t mail 8 mutt
```

```
screen -t daemon 9 tail -f /var/log/daemon.log
```

Ha ezeket a parancsokat a ***$HOME/.screenrc.multiwin*** állományba mentjük, a

```
# screen -c $HOME/.screenrc.multiwin
```

paranccsal rávehetjük a Screent, hogy a saját könyvtárban található ***screenrc*** állományunk helyett ezt használja.

Az indító héjprogramokból olyan Screen-munkamenetek is indíthatók, amelyek inkább a rendszer jellemzőit mutatják. A rendszer Screen-munkamenet gyakori alkalmazása a soros konzolkiszolgáló. A Screen kitűnő erre a célra, minthogy a soros terminálokhoz beépített támogatással és naplózással van felszerelve. Az alábbiakban egy ilyen célt szolgáló, megjegyzésekkel ellátott ***screenrc*** állományt mutatunk be:

```
# Ennek az ÆllomÆnynak a hasznÆlata azt
# elıfeltØtelezi, hogy a felhasznÆl
# rendelkezik a megfelelı jogosultsÆgokkal,
# hogy a soros kapukat hasznÆlja Øs a
# napl ÆllomÆnyokba  rjon


# kapcsoljuk be a napl zÆst az  sszes ablakra
# vonatkoz an
deflog on

# a screen napl ÆllomÆnya a
# /var/log/serial.$WINDOW legyen
logfile /var/log/serial.$WINDOW

# ablakok nyitÆsa a soros kapun
screen /dev/ttyS0 38400
screen /dev/ttyS1 19200
```

Ha ezeket a parancsokat a */etc/screenrc.serial* állományba mentjük, ezt a következő héjprogrammal lehet a Screen indulásakor bekapcsolni:

```
# su serialuser -c screen -dms serial -c
➥/etc/screen.serial
```

A `-dms serial` kapcsoló arra utasítja a Screent, hogy a munkamenetet leválasztott üzemmódban indítsa el, és „serial”-nak nevezze el. A soros kapu használatához jogosultsággal rendelkező felhasználó bejelentkezhet és csatlakozhat ehhez a munkamenethez, pontosan úgy, mint bármilyen másik közönséges Screen-munkamenethez. A Screen leválasztott módú indítása lehetővé teszi, hogy a Screent `cron` feladatként indítsuk el, amennyiben éppen erre van szükségünk. Lehetséges egyetlen, az egész rendszerre érvényes ***screenrc*** létrehozása, amelyhez több felhasználó csatlakozhat. A Screen az ablakokhoz kapcsolódó ACL-ek (Access Control List) révén támogatja a többfelhasználós üzemmódot, vagyis megszabja, hogy melyik felhasználó mit tehet és mit nem. A több felhasználó által alkalmazható Screen azonban `setuid root` üzemmódot igényel. Eme követelmény miatt többfelhasználós Screen-munkameneteket bemutató példákat szerintem nem célszerű taglalni. Ha mégis többfelhasználós Screen-munkamenetet szeretnél üzembe állítani, akkor alaposan tanulmányozd át a Screen leírását, tedd fel a „`setuid root` jogosultságok adása egy bonyolult programkódhoz” üldözési mániás kalapodat, és készülj fel a lehetőségek lehető legszorosabb korlátozására. A harmadik megoldás az, hogy az előző két példát összefésülöd, és az egész rendszert érintő felhasználói közreműködést igénylő (interaktív) programokat a ***screenrc***-n keresztül indítod el. Ennek remek kihasználása a Mutella – a konzolalapú Gnutella ügyfélprogram – elindítása a Screen indulásakor. A Screennel ezt el lehet indítani, alkalmanként kapcsolódni lehet hozzá, meg lehet tudni az állapotát, és lekérdezéseket is lehet vele futtatni.

A screen program a 44. CD Magazin/screen könyvtárában található.

Linux Journal 2003. január, 105. szám

Adam Lazur (adam@lazur.org)
Linux-szaktanácsadó, működési területe a beágyazott rendszerektől a Beowulf-telepekig terjed. Szabadidejében szeret magáról egyes szám harmadik személyben írogatni. Adam szívesen fogad a cikkel kapcsolatos leveleket.

KAPCSOLÓDÓ CÍMEK

- http://www.math.fu-berlin.de/~guckes/screen
- http://groups.yahoo.com/group/gnu-screen

Felhasználói felület fénysebességgel

Nem is olyan régen még a programozók rémálma volt a felhasználói felület elkészítése. A program sokszor már régóta működött, amikorra a programozó rászánta magát a kezelőfelület elkészítésére.

Ez mára némiképp megváltozott, ugyanis a fenti feladat egyszerűsítése érdekében gomba módra szaporodtak és ma is szaporodnak a különböző eszközkészletek (tool kits). Ebben a cikkben egy olyan eszközkészletet mutatok be röviden, ami méltánytalanul kevéssé ismert, holott sok előnyös tulajdonsága van, sok mindenben talán jobb, de biztosan más, mint a többiek.

A türelmetlenek számára előrebocsátva: az FLTK-ról (Fast Light Tool Kit) lesz szó. Előbb azonban ejtsünk néhány szót általában az eszközkészletekről. Ami közös bennük: mindegyik biztosít a programozó számára olyan elemeket, amelyek a felhasználói felületek összeállítására általánosan használatosak. A teljesség igénye nélkül a legelterjedtebbek: gombok, menük, ablakok, beviteli mezők, gördítősávok, eszközsávok. Ezeket használhatjuk a készlet által megadott formában, vagy ha az illesztőfelülete objektumalapú, akkor saját típust képezhetünk belőlük, és ezt alkalmazhatjuk a későbbiekben.

Nézzük tehát, milyen lehetősége van a programozónak, ha a „lényegi" munkát végző programjához egy kellemes, könnyen kezelhető felületet akar biztosítani. Mi az, amiben a fent említett elemek jelentős része benne van, mindenütt hozzáférhető, könnyen használható? Igen, a HTML alapú, böngészőben futó felületre gondolok! Alkalmazzák is a programozók elég széles körűen, mivel nagyon kevés munkával lehet működő felületet biztosítani. Előnyei: egyszerűség, rendszerfüggetlenség, hiszen bármely naprakész böngészőben működik, ily módon hordozható, végül: hálózati megoldásra is alkalmas. Szép számmal akadnak hátrányai is: a hatékony használathoz szükség van egy webkiszolgálóra, lassú, a programozó keze erősen meg van kötve a böngészők behatárolt és egymástól eltérő képességei miatt, pont akkor válik egyre kevésbé hordozhatóvá, ha olyan eszközöket használunk, amelyek bővítenék a felület képességeit, elsősorban a JavaScript-és a Java-betétekre gondolok.

Aki ennél hatékonyabb eszközkészletet szeretne, de mégis tart a „komoly" programozási nyelvektől, az feltehetőleg valamely parancsnyelvet és a hozzá illesztett eszközkészletet választja. A leggyakoribb a Perl/Tk kapcsolat. Itt, mivel ez a Perl nyelven keresztül egy teljes eszközkészletet tesz elérhetővé, szinte minden olyan lehetőségünk megvan, ami más készletek esetén létezik. Fő hátrány ebben az esetben a lassúság, valamint hogy a Perlnek és mellette a Tk-nak, valamint a hozzá kapcsolódó moduloknak mind a fejlesztői, mind a futtató gépen fenn kell lenniük, valamint hogy a forrás a parancs sajátságaként kényszerűen „nyílt". Ugyanez a Tk különben a Tcl nyelvhez illesztve is megtalálható, számomra kissé érthetetlenebbül, de biztosan akad, aki tud Tcl-ben dolgozni…

Más parancsnyelveket (PHP stb.) nagyvonalúan átugorva, rátérek az „igazi" programozási nyelvekre. Ez a legtöbb Linuxban dolgozó programozó számára a C, újabban a C++. Természetesen ezekhez a nyelvekhez is vannak eszközkészletek. Sok esetben az eszközkészletek valamely feladat megalkotására, valamely program megírásának a céljával, azt segítendő jönnek létre. Ilyen például a Gimp–Gtk. Ehhez közeli a KDE–Qt páros, ahol az ablakkezelő a Qt könyvtáraira épül, és ezt az eszközkészletet használva fejlesztik. Más irányból indulva ugyan, de windowsos felületen ilyen az MFC C++ készlet is. Azért más egy kicsit, mert itt előbb volt meg a rendszer alapja Pascal, később C nyelven, az eszközkészletet csak utóbb írták meg, immár elméleti szempontokat is figyelembe véve. Ez látszik is rajta. Helyben is vagyunk, ugyanis a két utóbbi példát tekintve két C++-eszközkészletet láthatunk, és ezek legfőbb hibáját a túlzott bonyolultságban, a túl magas szintű (elvonatkoztatási szintről van szó, nem minőségiről) elemkezelésben látom. Ezeket az eszközkészleteket használva egy-egy párbeszédablak elkészítéséhez valóban grafikus fejlesztőeszközre van szükség. Emellett eseménykezelésük is elég bonyolult és erőforrás-igényes.

A fentieket ***Bill Spitzak***, az FLTK eredeti fejlesztője már régen felismerte. Célul egy egyszerű, gyors, könnyen kezelhető, logikus felépítésű, szerény erőforrás-igényű eszközkészlet kifejlesztését tűzte ki. Ebben munkaadója, a Digital Domain egy ideig támogatta, így született meg az FLTK első változata. Azóta a hivatalos támogatás megszűnt, a programcsomag LGPL felhasználási szerződés alatt került kibocsátásra. Jelenleg egy önkéntes csapat fejleszti, akik Interneten keresztül tartják egymással a kapcsolatot. Nézzük meg a legfontosabb tulajdonságait:

- Az LGPL felhasználási szerződés hatálya alá tartozik, azzal a záradékkal, hogy a program statikusan is összeépíthető, anélkül, hogy a forrásszöveget nyilvánosságra kellene hozni. Így üzleti célú programokban is használható.
- C++-osztályok, könnyű memóriakezelés.
- A jelenleg megszokotthoz képest rendkívül kis méret, egy minden elemet tartalmazó program mérete is mindössze 600 kilobájt körül van, statikusan szerkesztve is!
- Moduláris felépítés, a könyvtárból csak azok a részek kerülnek a programba, amelyek szükségesek; nincsenek elemközti kapcsolatok.
- Linux- (Unix, X11), Windows-, Mac-változatban is elérhető, azaz ugyanaz a forrásállomány bármelyik operációs rendszerre lefordítható.
- GNU, Borland, Microsoft stb. fordítóprogramokkal is lefordítható.
- Egyszerű és hatékony eseménykezelés: az egyes elemek eseménykezelő függvényükben „nyilatkozhatnak", hogy szükséges-e számukra az esemény, és amennyiben nem, a rendszer továbbküldi azt a következőnek.
- Beépített OpenGL-támogatást tartalmaz.
- Interaktív, grafikus felhasználói felületszerkesztő, C++-kódot állít elő, ami utólag szerkeszthető.
- Gyors és hordozható 2D-s grafikus függvények.
- Kiváló, pdf formátumú leírással bír (sajnos csak angol nyelven).

```
include ../makeinclude
clean:
      -@ rm -f $(ALL) *.o core *~


# a lØnyeg...
nezoclass.o: nezoclass.H
nezo$(EXEEXT): nezo.o nezoclass.o
../lib/$(IMGLIBNAME)
      echo Linking $@...
      $(CXX) -I.. $(CPPFLAGS) $(CXXFLAGS)
      ➥nezo.o nezoclass.o -o $@
      ➥$(LINKFLTKIMG) $(LDLIBS)
```

```
#include <stdlib.h>
#include <stdio.h>
#include <FL/Fl.H>
#include <FL/Fl_File_Chooser.H>

#include "nezoclass.H"

Kep_Nez_Wnd *kep_nez_wnd=(Kep_Nez_Wnd*)0;
Fl_File_Chooser
*kep_opener=(Fl_File_Chooser*)0;


void init()
{
  fl_register_images();
  Fl_File_Chooser::add_favorites_label=
  ➥"HozzÆad a kedvencekhez";
...
...
}

int main(int argc, char ** argv) {
 init();
 ::kep_nez_wnd=new Kep_Nez_Wnd();
 ::kep_nez_wnd->loadkep("logo.png");
 ::kep_nez_wnd->show();
 Fl::run();
 }
```

Hogyan szerezhetjük be és bírhatjuk munkára? Ha a csomag nincs Linux-terjesztésünk lemezein, akkor megtalálhatjuk a 44. CD magazin/fltk könyvtárában, vagy töltsük le a ➲ http://www.fltk.org címről. Célszerű a legfrissebb üzembiztos változatot beszerezni. Ez a változat a cikk írásakor az 1.1.2-es volt. CVS-en keresztül a fejlesztői változat is elérhető, ez jelenleg a 2.0-s. Számos ígéretes újdonságot mutat fel, de egyelőre még befejezetlen. Az egész forrás másfél megabájtot tesz ki, tehát modemes kapcsolattal is kényelmesen letölthető. Ha megvan a csomag, bontsuk ki a

```
# tar xvzf fltk-1.1.2-source.tar.gz
```

paranccsal. Ekkor az ***fltk-1.1.2*** könyvtárba kerülnek az állományok. Ha fejlesztőeszközeinket megfelelően telepítettük, ebben a könyvtárban a

```
# ./configure
# make
```

parancsok kiadása után máris indul a fordítás. Ha a munka hibajelzés nélkül elkészült, a ***test*** könyvtárban találhatók a próba- és bemutatóprogramok, amelyeket ki is lehet próbálni. Ezeket az `xterm` ablakból indítsuk, ugyanis X alatt futnak, karakteres konzolról nem indulnak el.
Ha úgy gondoljuk, hogy később is használni akarjuk az FLTK-t, a

```
# make install
```

parancs segítségével telepíthetjük. Ez a lépés egyébként egyelőre szükségtelen.
Példaként egy olyan programot készítünk el, amely már ebben az egyszerű formájában is használható valamire. Nem szövegszerkesztőre gondolok, több okból sem. Először is: szövegszerkesztőkkel Dunát lehet rekeszteni, tehát nem hiányzik egy újabb; másodszor pedig az FLTK tartalmaz egy kész, egyszerű szövegszerkesztő elemet, amihez csak a kívánt menüket, gyorsbillentyűket, ikonokat kell hozzáadni, és már munkára is fogható. Ehelyett az alábbiakban egy egyszerű képnézőt írunk meg. Tapasztalatom szerint Linuxon futó jó képnézegető nem igazán van, talán az xv-t kivéve, így szükség is van rá.
Célkitűzésünk első megközelítésben a következő lesz:

- egyszerűség;
- könnyű kezelhetőség;
- bővíthetőség a további fejlesztések irányában (ki tudja, talán ez lesz „a” képnézegető Linuxon…);
- a legfontosabb: a megnyitott kép méretét igazítsa a programablakhoz. Roppant bosszantó, amikor állandóan tologatni kell, vagy menükben szükséges keresgélni a kicsinyítést;
- a legfontosabb szolgáltatások (nagyítás, kicsinyítés, ablakhoz illesztés) legyenek elérhetőek billentyűzetről is, mivel így sokkal könnyebb használni őket.

Az egyszerűség kedvéért, és mert így nem kell rendszerünkön véglegesen telepíteni az FLTK-t, akár egyszerű felhasználóként is megtehetünk minden lépést (ajánlott is). Az FLTK forrásfában hozzunk létre egy könyvtárat, párhuzamosan az src és test könyvtárakkal (azaz közvetlenül az ***fltk-1.1.2*** könyvtárban). Legyen a neve mondjuk ***nezo***. Mivel most „hagyományos” módon fogunk dolgozni, nem pedig grafikus, összevont fejlesztői környezetben, a kívánt fájlokat nekünk kell létrehoznunk. Előbb készítsük el őket üresen (`touch` vagy `> fÆjlnØv`):

- ***Makefile***
- ***nezo.cxx***
- ***nezoclass.cxx***
- ***nezoclass.H***

A fájlok elnevezésénél az FLTK megszokott elnevezési mintáját alkalmaztam, azaz a C++-fájlok ***.cxx***, a C++-fejállományok (include)***.H*** kiterjesztést kapnak.
(Az összes fájl, beleértve a kész bináris programot is, letölthető a cikk végén megadott címről, illetve megtalálható a 44. CD Magazin/fltk könyvtárában).
A `Makefile` fogja a GNU `make` programnak megmondani, mit is szeretnénk. Kedvenc szövegszerkesztőnkbe írjuk be az *1. listán* láthatókat (figyelem, ahol beljebb kezdődik a sor, ott TAB karakter van! Ha a Midnight Commander beépített szerkesztőjét használjuk, az ezt piros színnel jelzi is; a `make` érzékeny erre…).

Az utolsó rész egy sorba írandó. Az FLTK beállító parancsfájlja által fáradságosan elkészített `makeinclude` fájlt használjuk, mivel ez tartalmazza a rendszerünkre vonatkozó adatokat. Most pedig térjük rá a lényegre! Mivel ravasz módon a saját osztályokat külön fájlba helyezzük, a főprogram rendkívül egyszerű lesz (lásd *2. listán*).

Tekintsük át röviden a 2. listán látottakat! Minden FLTK programba be kell illeszteni a ***FL.H*** fejállományt. Ezenkívül pedig még azokat is, amelyek a használni kívánt elemeket tartalmazzák. Ebben az esetben ez az `Fl_File_Chooser`, azaz a fájlválasztó elem. Ez, mint működése során látni fogjuk, a szokásos (Windows és KDE alatt is) fájlkiválasztóhoz hasonlít, csak annyiban jobb azoknál, hogy lehetőséget biztosít kedvencek rögzítésére és szűrők megadására, amelyek a program következő indításakor is megtalálhatók lesznek.

Ezután beillesztjük a saját fejállományunkat (`#include "nezoclass.H"`). Ebben vannak az általunk használt osztályok, valamint az egyelőre egyetlen menü meghatározásai. Ekkor két teljes hatókörű változó, ez esetben két mutató megadása következik. Erre azért van szükség, hogy ezek a program minden részéből elérhetőek legyenek. Az egyik maga a főablak, a másik a fájlválasztó ablaka. Figyeljük meg a mutatók kezdeti `NULL` értékkel való feltöltésének módját: típuskényszerítéssel azt is megadjuk, hogy ezek milyen mutatók. Ez jó szokás, ugyanis olvashatóbbá teszi a programot. Soha ne hagyjunk mutatót érték nélkül! – ez ugyanis a legbiztosabb módja a `SIGSEGV` hiba (Windows alatt Segmentation fault) elérésének.

Egy függvény, az `init` következik. Ezt a `main` függvénybe is írhattuk volna, de így áttekinthetőbb. Az `fl_register_images` meghívása azért szükséges, mert az FLTK a különböző képformátumok kezelésekor úgynevezett kezelő (handler) módszert alkalmaz. Ez a függvény tölti be az összes, a programkönyvtárban fellelhető formátum kezelőjét. (Ehhez a gépünkön ***jpeglib***-nek és ***pnglib***-nek kell lennie a megfelelő formátumok kezeléséhez, amelyeket minden Linux-terjesztés tartalmazza). Az `init` további részében a fájlválasztót „vesszük rá” a magyar beszédre. Ha ezt kihagyjuk, akkor az alapértelmezett angol szöveggel működik.

Végül a főprogram a maga egyszerűségében: meghívja az `init`-et, létrehoz egy képnéző ablakot (`Kep_Nez_Wnd`), meghívja ennek képbetöltő tagfüggvényét, majd megjeleníti azt. Az utolsó sor jellegzetes `FLTK: Fl::run();`. Ezzel lép be a program az eseménykezelő ciklusba.

Miként tapasztaljuk, ettől még nem lettünk túl okosak. A lényegi rész nyilván a ***nezoclass*** fájlokban rejlik. Valóban így van. Nézzük meg a *3. listát*, és ott a ***nezoclass.H*** fájlt. A sorokat megszámoztam, hogy hivatkozni tudjak rájuk. A sorszám fordításkor ne legyen benne!

Az 1–5. sorban ismét a fejlécbeillesztések vannak. Itt már több elemet használunk. A ***Window*** maga az ablak, a ***Box*** egy olyan elem, amelyhez képet és szöveget is lehet rendelni, most ezt alkalmazzuk a kép megjelenítésére. A ***Scroll*** tetszőleges, más elemek görgetését teszi lehetővé, a Menu_Button alkalmazható lenyíló, és mint ebben az esetben tesszük, felbukkanó menük aktiválására; végül a legfontosabb a ***Shared_Image***. Ez teszi lehetővé, hogy tetszőleges, az FLTK által támogatott képet betöltsünk. Kiválasztja az adott típushoz tartozó kezelőt, és egy minden esetben azonos ***Image*** típust ad vissza, amit az FLTK programban már bárhol, ahol képet lehet beilleszteni, használni tudunk.

Ezután a 7–11. sorban található a `Kep_Box`, a képet majdan megjelenítő elemünk megadása. Két ok miatt hozzuk létre: egyrészt saját létrehozó függvényt (konstruktor) készítünk

```
1.  #include <FL/Fl_Window.H>
2.  #include <FL/Fl_Box.H>
3.  #include <FL/Fl_Scroll.H>
4.  #include <FL/Fl_Menu_Button.H>
5.  #include <FL/Fl_Shared_Image.H>
6.
7.  class Kep_Box : public Fl_Box{
8.      int handle(int event);
9.  public:
10.      Kep_Box(int,int,int,int);
11. };
12.
13. class Kep_Nez_Wnd : public Fl_Window{
14.     Fl_Window *wnd;
15.     int origw;
16.     int origh;
17.     float fact;
18.     float afact;
19.     Fl_Shared_Image* imag;
20.     Fl_Menu_Button* mb;
21. public:
22.     Kep_Nez_Wnd();
23.     void loadkep(const char *fname);
24.     void reloadkep();
25.     void zoomin();
26.     void zoomout();
27.     void zoomfit();
28.     Kep_Box *kephely;
29.     Fl_Scroll *oll;
30. };
```

neki, így a Box esetében a létrehozáskor nem kell nem alapértelmezett tulajdonságokat módosítani, másrészt saját eseménykezelő függvénye lesz (a handle), mivel azt szeretnénk, ha bizonyos billentyűeseményekre válaszolna. Azért nem maga az ablak fog eseménykezelőt kapni, mert akkor a `Scroll`-nak szóló eseményeket is nekünk kellene kezelnünk, ezt pedig nem akarjuk, mert azt jól tudja nélkülünk is.

A 12. sortól a fájl végéig a főablak megadása található. Mint látható, az FLTK ***Window*** eleméből származtatjuk, így csak az eltérő adat- és függvénytagokat kell meghatároznunk. Először néhány adattag: ezek fogják tárolni az eredeti képméreteket, a nagyítási tényezőt, magát a pillanatnyi képet, valamint a menügombelemet, illetve a végén nyilvánosként (public) a `Kep_Box` és a `Scroll` elem mutatói.

Végül az eljárások szerepelnek: a létrehozókon kívül a kép betöltését, nagyítás utáni újratöltését, és a nagyítást, illetve kicsinyítést lehetővé tevő függvények.

Nincs más hátra, mint hogy a *4. listán* megnézzük, hogyan is működnek ezek a függvények (ismét sorszámozva).

Az 1–8. sorban újra a már megszokott fejlécfájl-beillesztéseket láthatjuk, az `fl_ask` az egyszerű üzenet és döntés ablakok meghatározását tartalmazza. A 10–11. sorban adjuk meg, hogy a főprogramban meghatározott két mutató külső, azaz másik fájlban keresendő (extern). Majd egészen a 65. sorig az úgynevezett visszahívandó (callback) függvények következnek. Az FLTK-ban a különböző elemekhez hozzárendelhetők ilyen függvények, amiket akkor hív meg a rendszer, ha az adott elemet aktiváltuk, például menüpontnál kattintunk rajta. A különböző elemek esetében ez a függvény különböző események során kerül meghívásra. Egyelőre elég tudni, hogy a menüpontoknál

kattintáskor, illetve, ha gyorsbillentyű van hozzárendelve, akkor annak megnyomásakor történik ez. A mi meghívandó függvényeink mind a különböző menüpontokhoz tartoznak.
A következő éppen ez: a `menu_popup` tömb, amely `Fl_Menu_Item` (azaz menüpont) elemekből áll. Egy ilyen elem felépítése a következő:

```
(sz veg, gyorsbillentyß, megh vand  f ggvØny,
➥felhasznÆl adat, beÆll t  zÆszl k (flag))
```

valamint a végén még hozzá lehetne fűzni a megjelenítés betűtípusára vonatkozó adatokat. Már a beállító jelzőket sem kötelező megadni, ennek alapértelmezett értéke 0. Néhány menüpontnál nincs is, ahol van, ott most az `FL_MENU_DIVIDER` jelzőt használjuk, ennek hatására az FLTK az adott menüpont után egy elválasztó vonalat jelenít meg.
A másik gyakran használt jelző az `FL_MENU_INACTIVE`, amelynek esetén a menüpont nem lesz kiválasztható, ezért halványabban kerül megjelenítésre.
Következik a `Kep_Box` létrehozója, amelynél csak annyiban írjuk felül az `Fl_Box`-ot, hogy mindenféle keret nélküli megjelenítést írunk elő (`FL_NO_BOX`).
Érdekesebb viszont a `Kep_Box` eseménykezelő tagfüggvénye, amiben jelenleg a billentyűparancsokat kezeljük le. A számbillentyűk melletti + (összeadásjel) nagyít, a – (kivonásjel) kicsinyít, a * (csillag) pedig az ablakmérethez illeszti a képet. Csak az `FL_KEYUP` típusú esemény érdekel bennünket ebben az esetben. Azért nem az `FL_KEYDOWN` (lenyomás), mert azt – előttem még ismeretlen ok miatt – nem olyan megbízhatóan kapjuk meg. Figyeljük meg, hogy ha az adott elem felhasználja az eseményt, akkor igaz értéket (1-et) ad vissza, ha viszont nem, például ebben az esetben az összes többi gomb lenyomásakor, akkor hamisat (0), így a többi elem még felhasználhatja azokat. Ezzel a módszerrel oldja meg az FLTK az események láncolását, így nincs szükség a KDE esetén alkalmazott „érdekeltségi” táblákra, amelyek erősen növelik a program méretét, lassúságát és olvashatatlanságát.
Ha olyan eseményt kaptunk, ami egyáltalán nem érdekel bennünket, ebben az esetben a billentyűfelengedésen kívül mindent, akkor meghívjuk az alapértelmezett eseménykezelőt (`Fl_Widget::handle`), átadva neki az eseményt, hogy csináljon vele, amit akar (többnyire eldobja, de lehetnek bizonyos programszintű gyorsbillentyűk vagy nem ki és bevitellel kapcsolatos események, amiket kezel). Mint látható, itt ugyanazokat csinálja, mint a megfelelő menüpont által meghívott visszahívó függvények. Igen, valóban meg lehetett volna tenni, hogy ezeket a billentyűket az adott menüpontokhoz gyorsbillentyűként rendeljük hozzá, de akkor nem lenne eseménykezelőnk, és a további bővítéshez ez még hasznos lesz.
A figyelmes szemlélőnek feltűnhet, hogy nincs egérgombkattintáshoz rendelt eseménykezelés, holott a helyi menünek a jobb egérgombra kellene megjelennie. Ez azért van, mert ezt a kérdést elintézi az `Fl_Menu_Button` saját eseménykezelője. Mint később látni fogjuk, az `Fl_Menu_Button` elemet az ablak teljes méretére nagyítjuk, így a jobb gombbal bárhol kattintva a menü működik.
Itt van végre a főablak létrehozója! Kicsit egyedi, mivel most csak egy példány lesz belőle, így előbb ellenőrzi, hogy ez megvan-e (a teljes hatókörű mutatón keresztül, ha még nincs, akkor ez `NULL` – az előrelátásunknak hála. Ha nincs, létrehozza. Később, ha több ablakot akarunk, ezt kell kivennünk, hogy másikat készíthessen.)
Mivel ez az `Fl_Window` elemből származtatott, így oda kell gondolnunk a következő sort:

```
this=new Fl_Window(600,450,"KØpnØzı");
```

A szülő létrehozóját a megadott értékekkel hívja meg. Az FLTK elemelrendezésének jellegzetessége, hogy a `new` után addig, amíg az adott elem `end()` függvényét meg nem hívjuk, az összes új elem ennek a gyermeke lesz. (Természetesen csak akkor, ha ez lehetséges.) Csak a `Group` (csoport, tároló) típusból származtatott elemeknek van `end()` tagfüggvénye, így csak ezek tartalmazhatnak más elemeket. A pdf-leírásban megnézhetjük, hogy mely elemek származnak az `Fl_Group`-ból. Tehát nálunk az ablak tartalmaz egy `Fl_Scroll` elemet (görgető), és ez tartalmazza a képdobozt. Az ablakhoz rendelt még a `Menu_Button` is. Az `Fl_Scroll` természetesen szintén `Group` típusú, így az `end()` zárja le.
Még egy FLTK-érdekesség. Az elemek helyzetét és méretét pontértékben írjuk be. Nincs külön elrendezés- (layout handler) kezelő, ám mivel megadhatjuk, mely elemek méretezhetőek át, az ablak átméretezésekor történő változást rugalmasan irányíthatjuk. A mi esetünkben például ez oldja meg, hogy a `Menu_Button` mérete mindig megegyezzen az ablakéval, azaz mindenhol működjön.
A továbbiakban a többi tagfüggvényt határozzuk meg.
Ez elég egyértelmű. Ami érdekes, az talán a képbetöltés: a képfájlt az `Fl_Shared_Image` `get()` tagfüggvénye segítségével töltjük be. Ez a fájl tartalma alapján (nem csak a kiterjesztést nézi, tehát egy JPG fájlnak például ***.dat*** kiterjesztése is lehet, és ilyenkor is helyesen tölti be) kiválasztja a megfelelő kezelőt, majd egy képmutatót ad vissza. Figyelemre tarthat számot ez is: az új betöltés előtt a régi képet töröljük, hogy ne foglalja a memóriát. Mivel alapként mindig az `Fl_Shared_Image` szolgál, és ennek `copy()` tagfüggvényét használva másoljuk át szükség szerinti méretben a `Kep_Box`-ba, újabb kép betöltésekor ezt is törölni kell. Ezt nem tehetjük meg úgy, mint az `Fl_Image`-dzsel, azaz a delete paranccsal. Ezeket a képeket ugyanis az FLTK egy tömbben tárolja, mert ennek az elemtípusnak az is célja, hogy bizonyos képeket több helyen, újra lehessen használni, újabb betöltés nélkül (ezt mutatja a neve is). A `get()` csak akkor olvassa be újra a fájlból, ha még nincs a memóriában képként, így a `release()` tagfüggvényével kell törölni. Utána a megfelelő `NULL` értéket adjuk a mutatónak. Igaz, hogy pár sorral

```
1.   #include <FL/Fl.H>
2.   #include <FL/Fl_Box.H>
3.   #include <FL/Fl_Window.H>
4.   #include <FL/Fl_Shared_Image.H>
5.   #include <FL/Fl_Menu_Button.H>
6.   #include <FL/Fl_File_Chooser.H>
7.   #include <FL/fl_ask.H>
8.   #include "nezoclass.H"
9.
10.  extern Kep_Nez_Wnd* kep_nez_wnd;
11.  extern Fl_File_Chooser* kep_opener;
12.
13.  void exit_cb(Fl_Widget *, void *) {
14.    exit(0);
15.  }
16.
17.  void nevjegy_cb(Fl_Widget*w,void *d)
18.  {
19.   fl_message("FLTK KØpnØzı ver. 0.1\n
      ➥KØpnØzı dem  FLTK-hoz\n
      ➥(C)2002 HavrÆnek Ferenc\n
20.  Ez a program alkalmaz az Independent
     ➥JPEG Group Æltal kØsz tett
     ➥rØszeket.");
21.  }
22.
23.  void fit_cb(Fl_Widget *w,void *d)
24.  {
25.  Kep_Nez_Wnd *p;
26.  p=::kep_nez_wnd;
27.  p->zoomfit();
28.  p->reloadkep();
29.  p->redraw();
30.  }
31.
32.  void nagyit_cb(Fl_Widget *w,void *d)
33.  {
34.  Kep_Nez_Wnd *p;
35.  p=::kep_nez_wnd;
36.  p->zoomin();
37.  p->reloadkep();
38.  p->redraw();
39.  }
40.
41.  void kicsit_cb(Fl_Widget *w,void *d)
42.  {
43.  Kep_Nez_Wnd *p;
44.  p=::kep_nez_wnd;
45.  p->zoomout();
46.  p->reloadkep();
47.  p->redraw();
48.  }
49.
50.  void kep_open_cb(Fl_Widget *w, void *d)
51.  {
52.  if(!::kep_opener){
53.   Fl_File_Chooser *fc=new Fl_File_Chooser
      ➥(".","KØp fÆjlok (*.{bmp,gif,jpg,png,
      ➥pnm,pbm,BMP,GIF,JPG,PNG})",
      ➥Fl_File_Chooser::SINGLE,
      ➥"kØp bet ltØs");
54.  fc->preview(1);
55.  kep_opener=fc;
56.  kep_opener->newButton->hide();
57.  }
58.  kep_opener->rescan();
59.  kep_opener->show();
60.  while(kep_opener->visible()) Fl::wait();
61.  if(kep_opener->count()){
62.   ::kep_nez_wnd->loadkep
      ➥(kep_opener->value());
63.   ::kep_nez_wnd->redraw();
64.   }
65.  }
66.
67.  Fl_Menu_Item menu_popup[] = {
68.      {"KØp kivÆlasztÆsa", 0,
      ➥kep_open_cb,0,FL_MENU_DIVIDER},
69.      {"Passz t",0,fit_cb,0},
70.      {"Nagy t", 0, nagyit_cb, 0},
71.      {"Kicsiny t",0,kicsit_cb,
      ➥0,FL_MENU_DIVIDER},
72.      {"NØvjegy",0,nevjegy_cb,0,
      ➥FL_MENU_DIVIDER},
73.      {"KilØp",0,exit_cb,0},
75.      {0}
76.  };
77.
78.  Kep_Box::Kep_Box(int x,int y,int w,int h)
     ➥: Fl_Box(x,y,w,h)
79.  {
80.  this->box(FL_NO_BOX);
81.  }
82.
83.  int Kep_Box::handle(int event)
84.  {
85.  Kep_Nez_Wnd *p;
86.  p=::kep_nez_wnd;
87.    switch(event){
88.   case FL_KEYUP:
89.   switch(Fl::event_key()){
90.           case FL_KP+'+': p->zoomin();
91.          p->reloadkep();
92.                redraw();
93.                return 1;
94.           case FL_KP+'-': p->zoomout();
95.                p->reloadkep();
96.          p->redraw();
97.          redraw();
98.          return 1;
99.           case FL_KP+'*':
             ➥p->zoomfit();
100.               p->reloadkep();
101.               p->redraw();
102.               redraw();
103.               return 1;
104.  default: return 0;
105.  }
106.  default:  return
      ➥Fl_Widget::handle(event);
107.  }
```

folytatás a következő oldalon

folytatás az előző oldalról

```
108. }
109.
110.
111. Kep_Nez_Wnd::Kep_Nez_Wnd() :
     ➥Fl_Window(600,450,"KØp nØzı")
112. {
113.  if(!::kep_nez_wnd){
114.   mb=new Fl_Menu_Button(0,0,600,450,
       ➥"KØp men ");
115.   mb->type(Fl_Menu_Button::POPUP3);
116.   mb->menu(menu_popup);
117.
118.   oll=new Fl_Scroll(0,0,600,450);
119.   oll->type(Fl_Scroll::BOTH_ALWAYS);
120.   kephely=new Kep_Box(0,0,584,434);
121.   oll->end();
122.   this->end();
123.   afact=1.0;
124.   fact=1.2;
125.   imag=(Fl_Shared_Image*)0;
126.   ::kep_nez_wnd=this;
127.   this->resizable(mb);
128.   this->resizable(this);
129.   wnd=this;
130.  }
131. }
132.
134.
135. void Kep_Nez_Wnd::zoomin()
136. {
137.  afact=afact * fact;
138. }
139.
140. void Kep_Nez_Wnd::zoomout()
141. {
142.  afact=afact / fact;
143. }
144.
145. void Kep_Nez_Wnd::zoomfit()
146. {
147. float wd,hd;
148.  if(origw< w() && origh< h())
      ➥{ afact=1.0; return; }
149.  wd=(float)(w()-16) / (float)origw;
150.  hd=(float)(h()-16) / (float)origh;
152.  afact=(wd <= hd) ? wd : hd;
153.  if(afact>1.0)afact=1.0;
154.  this->oll->position(0,0);
155. }
156.
157.
158. void Kep_Nez_Wnd::loadkep
     ➥(const char *fname)
159. {
160.  Fl_Image *img;
161.  Fl_Image *oldimg;
162.  int nw,nh;
163.  afact=1.0;
164.  fact=1.2;
165.  if(imag) { imag->release();
      ➥imag=(Fl_Shared_Image*)0; }
166.  oldimg=this->kephely->image();
167.  if(oldimg){ delete oldimg;
      ➥this->kephely->image((Fl_Image*)0); }
168.  imag=Fl_Shared_Image::get(fname);
169.  origw=imag->w();
170.  origh=imag->h();
171.  zoomfit();
172.  nw=(int)(afact*origw);
173.
174.  nh=(int)(afact*origh);
175.  img=imag->copy(nw-1,nh-1);
176.  this->kephely->size(img->w(),img->h());
177.  this->kephely->image(img);
178. }
179.
180. void Kep_Nez_Wnd::reloadkep()
181. {
182.  int nw,nh;
183.  Fl_Image *img;
184.  Fl_Image *oldimg;
185.  oldimg=this->kephely->image();
186.  if(oldimg){ delete oldimg;
      this->kephely->image((Fl_Image*)0); }
187.  nw=(int)(afact*origw);
188.  nh=(int)(afact*origh);
189.  if(nw>=origw || nh>=origh){ nw=origw-1;
      ➥nh=origh-1; afact=1.0; }
190.  img=imag->copy(nw,nh);
191.  this->kephely->size(img->w(),img->h());
192.  this->kephely->image(img);
193. }
```

lejjebb már be is töltünk egy fájlt, és új értéke lesz, szóval értelmetlennek tűnik. De mi történik akkor, ha sikertelen a betöltés? Akkor a későbbiek folyamán egy érvénytelen mutató lenne a változóban, ami sok bajt okozhatna.
A `reload()` tagfüggvény a megfelelő méretszámítások után másolatot készít a tárolt `Fl_Shared_Image`-ről, és ezt hozzárendeli a `Kep_Box`-hoz, ami azután megjeleníti.
Röviden ennyit az FLTK-ról. Igyekeztem rámutatni, hogy érdemes megnézni. Akit mélyebben érdekel, javaslom, olvassa el a 44. CD Magazin/fltk/ könyvtárában lévő példaprogramok forrását, valamint az FLTK forrását. Rendkívül logikus és áttekinthető a felépítése. Aki félt a C++-tól, érdemes ezzel kezdenie, mert itt hamar sikerélményhez juthat!

Sorozatunk következő részben elkezdjük bővíteni a programot. Az első számú cél, hogy több fájlt lehessen betölteni egyszerre, akár több könyvtárból is, és a billentyűzet segítségével mozogni lehessen közöttük. Esetleg menteni lehessen az adott listát hogy a betöltés után önműködően le tudjuk játszani őket.

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.

Betekintés a Blender rejtelmeibe

Ezzel a bevezető leírással útjára indul a Blender bemutatását célzó sorozatunk.

Tartsanak velem, és ismerkedjenek meg önök is egy nagy tudású 3D-modellező és -animációkészítő programmal! A program segítségével megvalósíthatjuk a képzeletünkben eddig csak képekként élő ábrándjainkat, inverz kinematikával mozgatott karakteranimációkat készíthetünk, görbék által határolt felületekből álló tárgyakat (NURBS-modelleket), animációkat alkalmazhatunk háttérképként és felületi mintázatként, a szerkesztőablakban háttérként elhelyezett animáció segítségével valósághű mozgásmodellezést végezhetünk, és – amit eddig még egyetlen ingyenes modellezőprogramban sem láthattunk –, játékokat is készíthetünk. Mindezeket a lehetőségeket egy 1, 5 MB-os program képében kapjuk meg, ami igazán nem nagy helyigény a mai világban, amikor már egy irodai programcsomag is több száz megabájtot foglal el.

Ismerkedés a Blenderrel

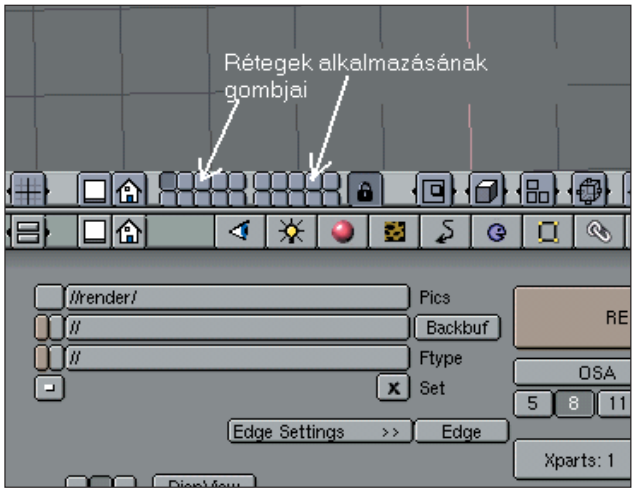
Animációs lehetőségei között megtalálható a hagyományos kulcskocka alapú megoldás; képes tárgyak egymásba alakítására (morphing), a mozgásokat pedig mozgásgörbékkel is megadhatjuk. A programmal képesek leszünk részecskerendszerek modellezésére, átlátszó tárgyak modellezésére, és az oly látványos lencsecsillanásról sem kell lemondanunk. A 3D-s tartalomfejlesztés sem elhanyagolandó képessége ennek a programnak, bárki könnyedén készíthet egyszerűbb játékokat vagy bemutatókat zenével és mindazzal, amit a Blender modellezés közben megvalósítani képes. A program két hátrányos tulajdonságát azonban nem hallgathatom el. Az egyik, hogy a képeket nem sugárkövetéssel számolja (ez valós idejű megjelenítésnél nem is igazi hátrány), tehát például nem tudunk majd valódi nagyítót készíteni, a másik pedig, hogy a vele készült játékokat futtatható formában csak a Blender Publisher segítségével jeleníthetjük meg, ennek az ára viszont egy kissé magas. Akit azonban ez nem zavar, annak bátran ajánlom a Blendert mind a szabadidő kellemes eltöltéséhez, mind pedig érdemi modellezéshez és animációkészítéshez, játékfejlesztéshez. Hogy ne csak beszéljünk, lássunk egy képet, amit véleményem szerint nem egy nap alatt készített az alkotó, de hűen tükrözi a program (és készítője) képességeit (*1. kép*).
A Blender elérhető Windows, Irix, FreeBSD, Mac OS X és Linux operációs rendszereken, de megjegyzem, hogy sorozatunkban természetesen a linuxos változatról lesz szó, egyrészt az újság jellege miatt, másrészt a linuxos változat megbízhatóbb működése következtében. Természetesen a Windows alatti Blender is használható, amennyiben valaki soha nem akarja ikonméretre váltani az ablakot. Eddigi tapasztalataim alapján ez a bonyolult művelet sok esetben a rendszer lefagyását eredményezi. Ha az olvasó által használt terjesztésnek nem lenne része a program, letölthető a ➲ http://www.blender.org valamint a ➲ http://www.pablosbrain.com/blender3d/ vagy a ➲ http://www.blender-forum.de/blender/bdownload.php címekről.
A programot elindítva rögtön szembetűnik a felhasználói felület egyszerűsége, ez azonban senkit ne tévesszen meg. A Blenderben rengeteg műveletet el tudunk a billentyűzet segítségével végezni, és már így, az ismerkedés elején javasolhatom, hogy az egyik kezünket tartsuk a billentyűzet közelében, ugyanis gyakorta szükségünk lesz a használatára. A Blender teljesen objektumközpontú, objektumosztályokat és -példányokat alkalmazva építi fel a jelenet leírására alkalmas rendszerszerkezetet. Ennek mélyebb megértésére azonban a későbbiek során lesz szükség. Addig elegendő lesz annyit megjegyezni, hogy minden objektumnak szinte minden értékét beállíthatjuk, és külső programból is megváltoztathatjuk.

1. kép Íme egy Blenderrel készült kép

A felhasználói felület testreszabása

A program telepítése és elindítása után a felhasználói felületet célszerű a saját ízlésünknek és igényeinknek megfelelően beállítani, hogy ezzel is növelhessük későbbi munkánk hatékonyságát. Mint az egy korszerű modellező programtól elvárható, a szerkesztőmezők és a nézetek elhelyezkedését szabadon beállíthatjuk. A program alapértelmezésben egy nézettel indul, a további nézeteket pedig a jobb egérgomb használatával hozhatjuk létre. Kattintsunk a jobb egérgombbal az alsó gombfelület és a szerkesztőmező között található keretre, és a megjelenő menüből válasszuk ki a ***Split area*** menüpontot.

A másik menüpont, a ***Join areas*** segítségével a felosztott területeket újra egyesíthetjük. Miután a nézetek megjelenítési módját igényeinknek megfelelően felosztottuk, az egyéni beállításokat a CTRL-U billentyűkkel tárolhatjuk. Személyes tanácsom, hogy elegendő két szerkesztőnézetet beállítani: az egyik legyen egy általános (bal, felső vagy elölnézet) nézet, míg a másikban a kamera képét látjuk. Ezeket a nézeteket természetesen a pillanatnyi igényeknek megfelelően változtathatjuk. A nézetek váltogatására a számbillentyűzet a legalkalmasabb eszköz.
A 3-as billentyűvel a jobb oldali nézetet érhetjük el, az 1-es billentyűvel az elölnézetet, míg a 7-es billentyűvel a felülnézetet. Mindhárom nézet 15 fokos lépésekkel elforgatható két tengely körül a 2-es, a 8-as valamint a 4-es és 6-os billentyűkkel. Az 5-ös billentyűvel a perspektivikus vetítést kapcsolhatjuk ki és be, a + (összeadásjel) és - (kivonásjel) billentyűkkel pedig a nagyítás mértékét állíthatjuk. A kamera nézetének kiválasztására a 0 billentyű használható. A nézeteket egér segítségével is eltolhatjuk, elforgathatjuk és nagyíthatjuk. A CTRL billentyű lenyomása és a középső egérgomb lenyomva tartása mellett mozgatott egérrel a nagyítást változtathatjuk, a SHIFT billentyű



2. kép Rétegek kezelése

és a középső egérgombos mozgatás a nézet látóterének eltolását eredményezi, míg ha csak a középső egérgombot tartjuk lenyomva és így mozgatjuk az egeret, térben forgathatjuk a nézet koordináta-rendszerét. Ha egy tárgyat úgy szeretnénk megjeleníteni az ablakban, hogy középen legyen, a bal egérgombbal helyezzük el a 3D kurzort a kívánt tárgy belsejében, majd nyomjuk meg a C billentyűt, és a nézetet a kívánt mértékben nagyítsuk. Ha minden tárgyat egyidejűleg szeretnénk egy nézetben megjeleníteni, akkor a SHIFT-C billentyűkombinációt kell használnunk.
Miután megismerkedtünk az alapvető műveletekkel, hozzunk létre egy egyszerű objektumot, és gyakoroljuk egy kicsit a szerkesztési műveleteket is.
Objektumok létrehozására a program menüjében nyílik lehetőségünk. Ezt a menüt a szóköz billentyűvel érhetjük el.
A menü bal oldali pontjaiból nyílnak az almenük, és ezek a pontok a menü szerkezetének megfelelően változnak majd. Tehát egy egyszerű gömb létrehozásához az ***Add*** menüpontból válasszuk ki a ***Mesh*** pontot, majd itt az ***UVSphere*** pontot. Ennek hatására megjelenik egy szerkesztőmező, amelyben a számra kattintva megváltoztathatjuk a létrehozandó tárgy tulajdonságait. A számtól balra kattintva az érték csökkenését érhetjük el, jobbra kattintva pedig a növelését. Pontos értéket is beírhatunk, ha a SHIFT billentyű egyidejű lenyomásával kattintunk a számon. Az ***UVSphere*** objektumnak két értéke létezik: a vízszintes és a függőleges felosztás értéke. Minél nagyobbak ezek az értékek, annál több háromszögből fog felépülni a tárgy. A szerkesztési műveletek elsajátításához nem szükséges túl bonyolult tárgyat használnunk, tehát ezeket az értékeket nyugodt szívvel akár nyolcra is csökkenthetjük. Miután mindkét értéket meghatároztuk, megjelenik egy sárga ponthalmaz, mely meglepő módon egy gömböt formáz.
A sárga ponthalmaz két dolgot is jelez. Egyrészt a tárgy pontszerkesztő módban látszik (ez a ponthalmaz), másrészt minden pontja ki van választva (ettől sárga). A pontok kiválasztását az A billentyű lenyomásával szüntessük meg. E gomb szolgáltatása a „Mindent kiválaszt” névvel illethető, vagyis ha nincs kiválasztva semmi, akkor mindent kiválaszt, ha pedig minden ki van választva, megszünteti a kiválasztott állapotot. Egy-egy pontot a jobb egérgomb lenyomásával tudunk kiválasztani, a kijelölt pontokhoz pedig a SHIFT lenyomása melletti jobb egérgombbal tudunk újabbakat hozzáadni. Természetesen az úgynevezett „gumikeretes” kijelölésre is lehetőség nyílik. A B billentyű lenyomásával egy célkereszt kerül elő, amivel a rajzprogramokban megszokott módon egy tetszőleges ponthalmazt választhatunk ki. A B billentyű ismételt lenyomásával egy kör alakú kijelölő mezőbe eső pontokat tudunk jelölni. A TAB billentyűvel tudunk a pontszerkesztő és a tárgyszerkesztő mód között váltani. Pontszerkesztő módban a tárgy minden pontját és élét külön-külön tudjuk változtatni, míg a másik módban a tárgyat egészében alakíthatjuk.

Alapvető átalakulások

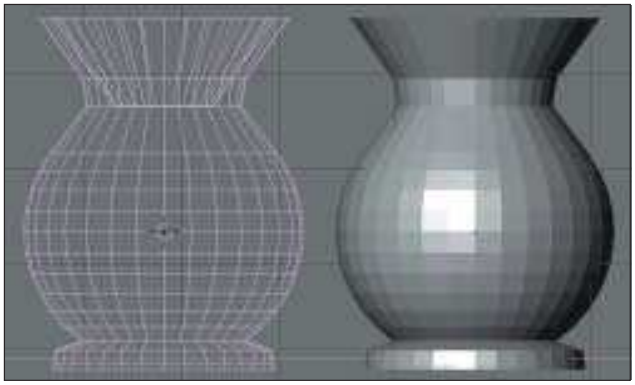
Tekintsük át, hogy az alapvető átalakításokat – az eltolást, az elforgatást és a méretezést – hogyan tudjuk a tárgyakon végrehajtani. A tárgy vagy tárgyak kijelölése után több lehetőségünk is van a változtatások végrehajtására. Az első és leggyorsabb megoldás, ha a bal egérgomb lenyomása mellett egy egyenest vagy körívet rajzolunk a nézetben. Ha egyenest rajzoltunk, a gomb felengedése után a tárgyat eltolhatjuk, majd a bal egérgomb lenyomásával új helyzetben rögzíthetjük. Amennyiben körívet rajzoltunk, a fenti műveletsor alkalmazása a tárgy elfordulását fogja eredményezni. A másik lehetőségünk az eltolásra, hogy a kijelölés után a tárgyat a jobb egérgombbal mozgatjuk. Az eltolást a G billentyű lenyomásával is aktiválhatjuk. A forgatásnak megfelelő billentyű az R. Az átméretezést billentyűzet segítségével kezdeményezhetjük, mégpedig az S billentyű lenyomásával. Mindhárom alapműveletet a SHIFT lenyomva tartásával finomíthatjuk, így ugyanis pontosabb mozgást érhetünk el. A CTRL gomb lenyomása mellett a tárgyat egységnyi lépésekben mozgathatjuk, vagy 5 fokonként forgathatjuk el. Természetesen a Blender lehetőséget nyújt arra is, hogy a tárgyainkat ne kizárólag szabadkézzel alakítsuk. Ha egy művelet megkezdése (a billentyű lenyomása) után a középső egérgombot nyomva tartjuk, akkor az eltolás, forgatás vagy átméretezés csak az adott nézetben egy tengely mentén érvényesül. Ezt a tengelyt az határozza meg, hogy melyik irányban mozdítjuk el az egeret a gomb lenyomása és a középső egérgomb lenyomása közötti időszakban. Ha tehát egy tárgyat csak a nézetben érvényes vízszintes tengely mentén szeretnénk eltolni, a G billentyű lenyomása után az egeret mozdítsuk el vízszintesen, majd a lenyomott középső egérgomb nyomva tartása mellett mozgassuk a kívánt helyzetbe.
Gyakran szükség lehet egy-egy elkészült tárgy másolatára, ha például egy objektum tengelyesen szimmetrikus, elegendő

Nézetek felosztása	Jobb egér a kereten, majd „Split area”
Nézetek egyesítése	Jobb egér a kereten, majd „Join areas”
Nézetek mentése	Ctrl-U
Oldalnézet	A számbillentyűzeten 3
Elölnézet	A számbillentyűzeten 1
Felülnézet	A számbillentyűzeten 7
Kamera	A számbillentyűzeten 0
Perspektíva be/ki	A számbillentyűzeten 5
Nézet forgatása	A számbillentyűzeten 8–2 és 4–6
Kurzor a nézet közepére	C
Mindent megjelenít	Shift-C
Főmenü	Szóköz
Eltolás	G
Forgatás	R
Méretezés	S
Tükrözés	S, majd X vagy Y
Másolat	Shift-D
Kijelölés	Jobb egérgomb
Összeadó kijelölés	Shift-jobb egérgomb
Kijelölés gumikerettel	B
Kijelölés köralakban	B-B
Mindent kijelöl/ Kijelölést megszüntet	A

csak az egyik felét részletesen kidolgozni, majd egy másolatát tükrözve a másik oldalát is létrehozhatjuk. Ezt a műveletsort a SHIFT-D, majd az S és az X vagy Y billentyűkkel végezhetjük el. A SHIFT-D elkészíti a másolatot; ezt más esetekben is használhatjuk. Az S az átméretezést kezdeményezi, amit ha az X-szel együtt használunk, vízszintes tükrözést kapunk, az Y billentyűvel együtt használva pedig egy a függőleges tengely mentén tükrözött tárgy lesz a végeredmény.

Rétegek kezelése

A Blender egyik előnyös tulajdonsága a többi szabadon elérhető modellezőprogramokkal szemben a rétegek kezelésének lehetősége. Az 1. képen látható, hogy minden nézetben húsz réteg érhető el, amik között tárgyakat tudunk mozgatni, vagy el is tüntethetjük az éppen nem szükséges tárgyakat, így csökkentve a bonyolult jelenetek megjelenítéséhez szükséges időt. Az egyes rétegeket a nekik megfelelő gombra kattintva tehetjük elérhetővé, több réteget pedig a SHIFT billentyű lenyomása mellett jelölhetünk ki. A kijelölt tárgyakat a SHIFT-M billentyűkkel mozgathatjuk át egy másik rétegre.
Ennyi alapozás után foglaljuk össze az eddig tanult billentyűket és a velük elérhető hatásokat (lásd a táblázatot). Ennyi tudás már elegendő egyszerű tárgyak létrehozásához, azonban az itt következő feladat megoldásához meg kell még említenem az objektumok és háromdimenziós pontok törlésére használatos billentyűkombinációt is. Miután a törlendő részt kijelöltük, az X billentyűvel eltüntethetjük. Pontszerkesztő



3. kép Első modellünk

módban a törlés vonatkozhat pontokra, élekre és teljes háromszögekre, míg tárgyszerkesztés módban csak a teljes tárgyra. Gyakorlati feladatként hozzunk létre egy váza alakú objektumot. Célszerű kiindulási alap lehet egy ***UVSphere***, melynek felosztása legyen 32 szegmens és 20 vízszintes kör. A gömb létrejötte után pontszerkesztő módban vagyunk, és nincs más dolgunk, mint a gömbből kialakítani a váza formáját. Kapcsoljunk oldalnézetbe, és jelöljük ki a gömb tetején lévő köröket a B billentyű segítségével. Ezeket az Z tengellyel párhuzamosan toljuk el felfelé (a középső egérgomb alkalmazásával). Majd ugyanezzel a módszerrel, a többi kör eltolásával és megfelelő méretezésével alakítsuk ki a váza végső formáját. A gömb tetejét alkotó köröket süllyesszük be a gömb belsejébe, ezek alkotják majd a váza belső falát. Ezt nem szükséges a tárgy teljes mélységében kialakítani, hiszen általában nem látunk bele egy vázába. Miután a belső falat a szükséges mélységben kialakítottuk, már csak arról kell gondoskodnunk, hogy a vázába bele is tudjuk majd tenni a virágokat, vagyis ki kell lyukasztani. Erre alkalmas az előbb említett törlés művelet. Jelöljük ki a korábbi gömb legfelső pontját, ami most a váza belsejében található, és nyomjuk meg az X billentyűt. A megjelenő menüből válasszuk a ***Vertices*** menüpontot. Ezzel már készen is van az első modellünk, amelyhez már csak az anyagot és mintázatot kell hozzárendelnünk a sorozat további részeinek útmutatása alapján.
A fent leírt módszerrel készült váza a *3. képen* látható.
A sorozat következő részeiben már a fenti alapismeretekre építve bonyolultabb tárgyakat modellezünk, megtanuljuk az anyagok és felületi mintázatok készítésének lépéseit és lehetőségeit, továbbá az UV-térképek készítésének mikéntjét. Röviden be szeretném majd mutatni azokat a lehetőségeket, amelyeket a Blender használata ad a játékfejlesztés vonatkozásában, továbbá az animáció készítésének menetét és többféle módját. A játékok készítésénél megvizsgáljuk majd a Blender beépített 3D-s motorjának lehetőségeit és a hagyományos módszerrel (modellezés, animációk 3D modellezővel, 3D-motor hagyományos programnyelven) megvalósítható játékok fejlesztésének a lépéseit. Tartsanak velem, kedves olvasóim, és ismerkedjenek meg a 3D-modellezéssel e nagyszerű programon keresztül!

Fábián Zoltán (dzooli@freemail.hu, dzooli@yahoo.com)
25 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.

A Linux és a Star Trek

A Digital Domain Oscar-díjas programot használ Linuxon a Star Trek Nemesis és más filmek digitális hatásainak megalkotásához.

A Linux először akkor tört be Hollywoodba, amikor a kaliforniai Venice-ben működő Digital Domain (D2) stúdió a ***Titanic*** című film különleges hatásainak leképezéséhez (effect rendering) a GNU/Linuxot használt. A D2 munkatársaival akkor beszélgettünk, amikor a tervek szerint 2002. december 13-án bemutatandó ***Star Trek Nemesis*** című filmen dolgoztak, méghozzá Linuxszal. A D2 Linuxot használ mind a leképezőtelepén, mind a grafikusok asztali gépein is.

A D2 eddig 21 filmjének elkészítéséhez használta kedvenc rendszerünket, ezek között olyan alkotásokhoz is, mint a Titanic vagy a ***Csodás álmok jönnek*** című filmek, amelyek elnyerték a legjobb látványhatásokért járó Oscar-díjat. A D2 két Tudományos és technikai teljesítményért járó filmakadémiai díjat is nyert, az egyiket Track mozgáskövető programjáért, a másikat pedig a Nuke nevű szerkesztőprogramért.

A legtöbb stúdióhoz hasonlóan eredetileg a D2 is SGI számítógépeken futtatott SGI Irix és Unix operációs rendszereket használt a leképezőtelepén és a grafikusok munkaállomásain. Az 1996-os ***Dante pokla*** című filmmel végzett próba azt bizonyította, hogy a Linuxra való átállás keresztülvihető. „Először a linuxos leképezőtelep állt munkába" – emlékszik vissza ***Judith Crow***, a D2 termelési igazgatója. „A Titanic munkáit az Areté nevű cég Renderworld nevű tengerszimulációs programjával végeztük. A program háromszor olyan gyorsan futott a linuxos Alpha gépeinken, mint az Irix SGI számítógépeken."

A leképezőtelep által kitaposott úton a Linuxot a Nuke és Houdini programok jutatták el az asztali gépekig.

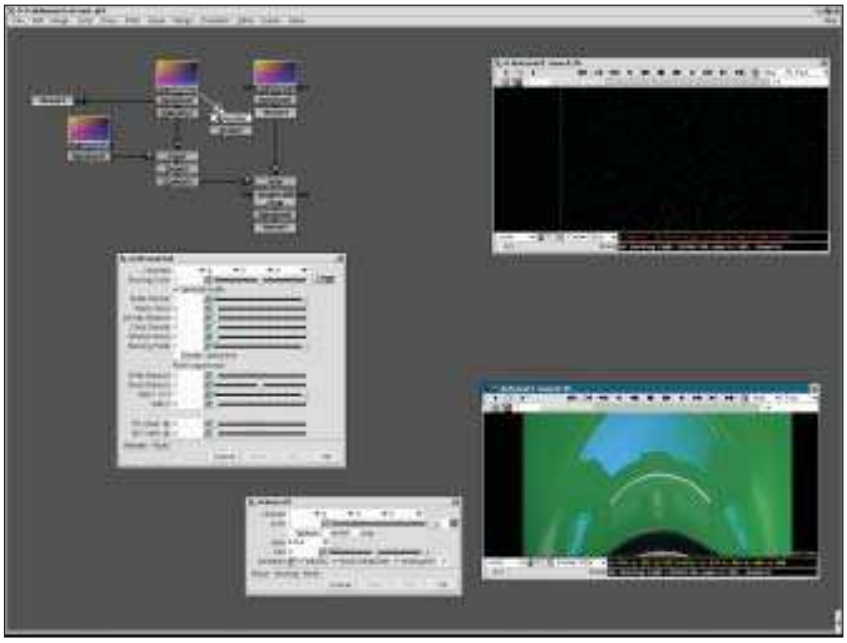
A szerkesztőprogramot (compositor) a szerkesztőgrafikusok használják a mozgóképek összehozására, például akkor, amikor az Enterprise űrhajó a háttérben lévő űrállomás előtt halad el. „A Digital Domain 1997 óta használja a Nuke-ot, vagyis mióta a Titanic megalkotásakor olyan széles körű szerepet kapott" – mondja ***Jonathan Egstad***, a Digitális hatások vezetője. Egstad a D2 további három munkatársával, ***Bill Spitzak***-kal, ***Paul Van Camp***-pel és ***Price Pethel***-lel a Nuke szerkesztőprogramért Oscar-díjat kapott.

„A Nuke lényegében kétdimenziós leképezőprogram" – tájékoztat Egstad. „Ötször-hatszor olyan gyors Linuxon, mint Irix alatt, de csak 2001 óta, mióta a Linux GUI gyors működésre is képessé vált. „Sok olyan eset van, amikor a kétdimenziós megoldás támogathatja a munkafolyamatot – mutat rá Egstad. – Létrehozhatunk egy teljes háromdimenziós jelenetet a Nuke-ban, majd hivatkozhatunk rá a 3D-s programban, mint amilyen a Maya, és megfordítva. A háromdimenziós jelenet létrehozható és leképezhető a Nuke3-ban, világítással, felületi mintákkal és árnyékokkal ellátva – a fényszórás, a Blinn és Phong hatások a program részét képezik. A kétdimenziós csomagok egyre inkább háromdimenziósakká válnak."

A Houdini kereskedelmi 3D-s programcsomag, amelynek egyik nagy felhasználója a D2 – legfrissebb változatában a Halo nevű saját, beépített szerkesztőjét kínálja. A Nuke-hoz hasonlóan ez is rangsoralapú (program), megfelelően kapcsolódva a 2D-s rendszerhez. A D2 emellett a LightWave és Maya nevű kereskedelmi 3D-s programcsomagokat is használja.

A Nuke harmadik változatát a D2 2001 óta használja Linux, Irix és Windows operációs rendszereken. A Nuke tervezésekor megőrizték az Irixen használt billentyűkombinációkat, így a felhasználóknak – különös tekintettel a D2-nél dolgozó szabadúszókra – nem kell tanulással tölteniük az idejüket, ha operációs rendszert kell váltaniuk. „A Nuke felhasználói felülete a további kiegészítő munkálatokra készülve tudatosan lett spártai" – jegyzi meg Egstad. – „Színkorrekciós eszközei talán az összes csomag közül neki a legjobbak."

A D2-höz évekig érkeztek a Nuke kereskedelmi forgalomba hozatalát célzó kérések, ami lehetővé tette volna a többi stúdió számára a használatát. Ez a nyomás még jobban felerősödött, miután az Apple felvásárolta a piacvezető Shake-et. A stúdiók elbizonytalanodtak,



1. kép Előkészületek egy csillagmező képének egy űrhajó ablakára történő beillesztésére a Nuke használatával. Az oldalt látható faszerkezet a szerkesztő parancsállomány csomóponti nézete

amikor az Apple késlekedett a jövőbeli Linux-támogatás bejelentését illetően. „A Nuke és egyéb, a stúdióban lévő vagy tervezett programok kereskedelmi forgalomba hozatala és árusítása céljából megalapítottuk a D2 Software Company-t" – meséli ***Michael Taylor***, digitális technikai igazgató – „és helyet biztosítunk a Nuke kipróbálására. A Nuke-nak azt a legújabb változatát kínáljuk, amelyet mi is használunk. Körülbelül két évvel ezelőtt, amikor a két- és háromdimenziós használat egyesítése érdekében a Nuke teljes újraírását határoztuk el, megfontolás tárgyává tettük a Shakere való átállást is, de végül úgy láttuk, a mi programunk jobb."

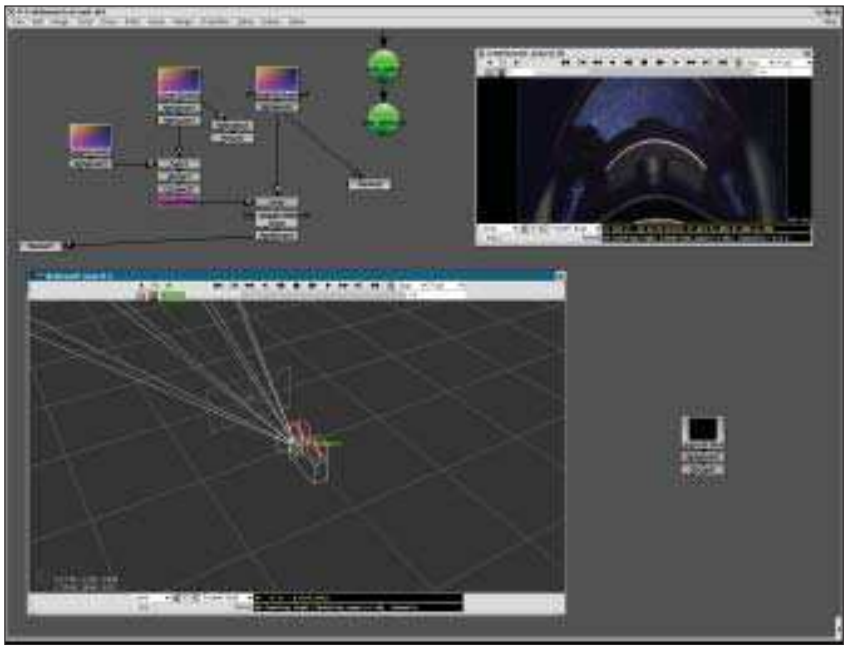
Taylor a Linux, Windows és Irix operációs rendszerek alatt működő változat megjelenését 2003 elejére ígéri. A Mac OS X rendszerrel kapcsolatban még nincsenek terveik. Az ára tízezer dollárnál kezdődik, ami hozzávetőleg a Shake árának felel meg. Diákok számára a Maya és Houdini oktatóváltozatához hasonlóan egy ingyenes vagy kedvező árú változat is elérhető lesz.

Brian Begun, aki digitális szerkesztést végez a cégnél, a Star Trek Nemesis egyik jelenetén mutatja be, hogyan folyik a munka a Nuke-kal:

„Egy átmeneti állományon dolgozom – ez egy olyan felvételsorozat, ami még nincs teljesen kész ahhoz, hogy a filmben elhelyezzük. Egy fejlesztői intranetet használunk minden egyes munkánk számára, ahol az egyes felvételek külön weboldallal rendelkeznek. Grafikusaink ezen keresztül osztják meg az adatokat. A jelenleg működő rendszer a Netscape-et használja rengeteg HTML-oldallal és egy Perl nyelven megírt kiszolgálóval. A könyvtárakba elhelyezett fájlok helyett inkább hivatkozásokat tárolunk, így lehetőségünk nyílik arra, hogy az adott állományt bármely lemezes tároló bármely könyvtárában anélkül tárolhassuk, hogy tényleges fizikai elhelyezkedésével foglalkoznunk kellene. Ilyen módon a rendszereinket karbantartó részleg szabadon bánhat a rendelkezésre álló tárterülettel, és anélkül növelheti a hatékonyságot, hogy az a fejlesztőmunkát befolyásolná."

Begun végigvezet minket a Nuke egy általános hatásának beállításán: az Enterprise űrhajót éppen egy csillagmezőn viszi át. Ez a Trek munkakörnyezete. Minden látvány a változók előre megadott listájával rendelkezik, válasszuk például a Star Trek SS145A-t:

```
$ job trek      [a lÆtvÆny
➥vÆltoz inak beÆll tÆsa]
```



2. kép A Nuke ablaka egy háromdimenziós drótvázas OpenGL-nézetet mutat a kamera szemszögéből

3. kép A végső simítások egy zöld képernyőn a Nuke-ban: az űrhajó pilótafülkéjének ablakára felkerülnek a csillagok

```
$ shot ss145a    [a
➥feladatvÆltoz k beÆll tÆsa]
```

A `cs` parancs a munkakönyvtárra vált, ami ebben az esetben a ***work.begun***:

```
$ cs
```

Innen átválthatok egy képeket tartalmazó könyvtárra, amely az alkotóelemeket és a munkaanyag egyéb részeit tartalmazza, vagy a munkakönyvtárra, amelyben a Nuke parancsfájljai foglalnak helyet, a felvevőgép mozgása esetén pedig a házon belüli Track-fájlok.

A munkakönyvtárban a Nuke, Flame, Track és Elastic Reality programok fájljait találhatjuk. (Az Elastic Reality egy viszonylag régi, de olcsó program, amely olyan roto- és Avid-képátalakítások létrehozására alkalmas, mint a rossz kép vagy a vezeték eltávolítása közbülső kép beszúrása segítségével.)

Ha a saját munkakönyvtáramat szeretném létrehozni, a `jsmk` parancsot használhatom. A többi könyvtár – például a képkönyvtár – létrehozása is ilyen mó-

don történhet. Ezek tartalmazni fogják az összes zöld képernyőt (green screen), a teljes felbontású és csökkentett felbontású helyettesítő képeket, az előzetes nézeteket, az ideiglenes kompozíciókat (amelyek az ügyfélnek a jelenet nagy vonalakban elkészített, vagyis nem feltétlenül kidolgozott vázlatát mutatják). Az `lss` parancs az `ls` eredményénél egy kicsit olvashatóbb formában mutatja az állományokat. Például ahelyett, hogy ilyen formában jelennének meg a fájlok:

```
test.0001.rgb
test.0002.rgb
test.0003.rgb
```

az `lss` így mutatja mindezt:

```
test.%04d.rgb 1-3
```

A Nuke futtatása előtt munkakönyvtáram Nuke-alkönyvtárára váltottam:

```
$ cs
$ cd nuke
$ nuke3
```

Amikor elindítjuk a Nuke-ot, egy grafikus ablak jelenik meg, amelyben az ***Image/Read/File*** menüpontot választottam, majd az előtérben lévő zöldképernyő-képek betöltéséhez az `ss145.wh` fájlt. Mialatt egy projekten dolgoztam, mind a nagyfelbontású, mind pedig a negyedakkora felbontású helyettesítő képeket használtam.

A képek Cineon 10 bit log formátumúak, amit a Nuke 16-bites lebegőpontos formára alakít át. A Nuke egy nézőkében legföljebb tíz kép egyidejű megjelenítésére képes. A billentyűzet számbillentyűit használva a tíz nézet bármelyike egyszerűen előhívható.

Íme egy pilótafülke zöld képernyője (lásd a *3. képet*). Amikor zöld képernyőt alkalmazunk, általában háromféle maszkot használunk. Az élmaszk célja az, hogy a fotó minden kis részletét megőrizzük. A kitöltő „innie" maszkot arra használjuk, hogy befedjük az esetlegesen látható lyukakat, amiket az előtérben lévő objektum előtti zöld foltok vagy anyagok okozhatnak. A tisztító (outtie) maszk a háttér nem odaillő objektumainak – például a színpadi lámpáknak – az eltávolítására szolgál. A maszkok alkalmazásához azt a ***backing color*** (háttérszín) szolgáltatást választom ki az ***Ultimatte*** színpipettájával, ami az eltávolítandó színhez a legközelebb áll – ez adja majd a legjobb maszkot. Ezután elvégzem a további szükséges átalakításokat, más maszkok alkalmazását és a letisztázást.

Jason Iversen műszaki igazgató a Star Trek energiahatásainak és a roncsoknak a megjelenítéséért felelős.

„A hajók robbanásaihoz a lehető legtöbb gyakorlatban alkalmazható hatást használtuk fel. Ezek rövidebb idő alatt megvalósíthatók, annak ellenére, hogy meg kell építeni a modellt, ami két ember kéthavi munkáját is igénybe veheti. Ugyanez háromdimenziós hatással viszont három ember négy-öthavi munkájába is belekerülhet. A robbanást 300 fps sebességű lassított felvétellel rögzítettük. Ez nem könnyű feladat és lehet, hogy nem is kelt valós hatást. Néhány robbanást a Houdini digitális roncsdarabjaival fejlesztettünk tovább. Az Enterprise pár lövése még valóságos, de a hajótestet elérő gyönyörű lövések már nem."

Beszélgetésünk alatt éppen az egyik SGI-gépet viszik el a leképezőtelepen való használatra. A D2 munkaállomásait kétprocesszoros pentiumos rendszerekre fejlesztik tovább.

A D2 a Star Trekkel kapcsolatos munkát korábban teljes egészében Linuxon, a Houdini használatával végezte, de a Maya grafikusainak nagy része Windows NT-t használ, mivel a Maya bővítményei Linux alatt nem elérhetők el.

„Legnagyobb képsorunk egy lavinamodellező szekvencia, amely a linuxos Houdinira és saját VoxelB nevű programunkra épül; ez végzi a térfogatelemzést" – tájékoztat Inversen, majd így folytatja: „A lavina egy hatalmas porszerű sáv, amely egészen háromdimenziós megvalósítást nyert, nem holmi kétdimenziós szemfényvesztés. Voxel-szerkesztőnk, a VoxelB tulajdonképpen egy bővítmény. Minden eszközünk képes adatok beolvasására a Mayából vagy a Houdiniból. Ezeket – amikor ömlő vizet akarunk előállítani – gyakran saját fejlesztésű folyadékdinamikai programunkkal kombináljuk. A Terragen a saját tereplétrehozó programunk, amelyet ***Az időgép*** című filmben a bolygók képeinél használtunk" – mondja Iversen, majd hozzáteszi: „Az előzetes képek előállítására és a digitális rajzolásokat végző munkatársaink számára készülő kezdeti ábrák létrehozásakor használjuk. A rajzolók mindegyike a Mayával dolgozik, elsősorban NT-n. A munkafolyamat ezen az előzetes képen alapul, ez kerül be az átalakítások után az alkotásba. A grafikusok mindegyike előzetes látványterveket állít össze, amelyek a szerkesztőkhöz kerülnek, akik a filmre alkalmazzák őket.

Bár a Linux támogatja a Houdinihez és a Mayához hasonló népszerű 3D-s programcsomagokat, Crow mégis úgy érzi, szenvednek a linuxos rajzolócsomagok hiányától: „Létezik a Photoshopnak olyan mélysége, ameddig a FilmGimp nem ér el, ugyanis a FilmGimp még nem elég érett hozzá." Crow elmondta még, hogy az Amazon16 ígéretes fejlesztés, egy 16-bites festőprogram, amelyet készítője, az Inertactive Effects Linuxra ültet át. Az Amazonnak komoly előélete van Irixen. „Már a Photoshop előtt is használta a rétegeket, támogatja a felhasználói makrókat, háromdimenziós felületminta-készítő képességekkel rendelkezik, és ami a legfontosabb, a Cineonhoz hasonlóan támogatja a HDR formátumot, ami nélkülözhetetlen a filmkészítésben – mondja Crow.

– A másik reményteljes 32-bites linuxos festőprogram-fejlesztés az Idruna által fejlesztett Photogenics, amely jelenleg még csak próbaállapotban létezik."

Crow szerint a D2 Irix alapú alkalmazásainak Linuxra való átültetése gyorsan lezajlott, s különösen igaz ez a saját fejlesztésű Nuke-ra. A D2-nél a Linuxra történő átállás szakaszosan zajlott: először a leképezőtelepre került sor, amely a hatások kötegelt feldolgozását végzi. „Amikor a Linux készen állt az asztali gépeken való alkalmazásra, lelkesen álltunk át a használatára. – meséli Crow. – Mihelyt úgy éreztük, hogy egy olyan operációs rendszer, mint a Linux támogatja azokat a tulajdonságokat, amelyekre építenünk kell, készek voltunk az átállásra."

Linux Journal 2003. január, 105. szám

Robin Rowe
(robin.rowe@movieeditor.com)
A Movie Editor.com internetes és televíziós videoalkalmazásokat készítő cég egyik partnere.

KAPCSOLÓDÓ CÍMEK

Amazon16 ➲ http://www.ifx.com/pages/amazon/16bit.html
Areté ➲ http://www.areteis.com
DigitalScape ➲ http://www.digitalscape.com
FLTK ➲ http://www.fltk.org
Idruna Software ➲ http://www.idruna.com
Nuke ➲ http://www.d2software.com

A PostgreSQL adatbázis-kezelő használata (2. rész)

Sorozatunk előző részében az eljárások készítéséig jutottunk el, most az ügyféloldali programok bemutatásával folytatjuk.

A PG négyféle C/C++ API-felülettel rendelkezik. Ez a gazdag lehetőség azt az igényt tükrözi, hogy a különféle feladatokhoz és programozói elvárásokhoz másmás eszközkészlet vezethet a legkönnyebben megoldáshoz. A legfontosabb és legalapvetőbb API a „natív" C alapú programozói felület, amit gyakran „PQ" API-nak is hívnak, utalva arra, hogy ennek az eszközkészletnek a megvalósítása a ***libpq.a***c és ***libpq.so*** könyvtárakban található. A példaprogramban érdemes észrevenni azt, hogy bár a PQ API elemi szintű eszközkészlet, mégis könnyű programozni.

A ***PQ*** könyvtár mellett létezik egy egyszerűsített, PGEASY nevű API (ennek helye a ***libpgeasy.a*** és a ***lobpgeasy.so*** fájl), melynek megvalósítása során az volt a cél, hogy amit csak lehet, az API függvényei a háttérben önműködően kezeljenek. Ez azt a hátrányt is jelenti, hogy a programozó nem férhet hozzá könnyen olyan adatokhoz, mint például a munkamenetleíró, ugyanakkor az alkalmazói program sokat egyszerűsödhet.

A C++ adta absztrakciós lehetőségeket aknázza ki a harmadik, PQ++ nevű felület (ennek helye a ***libpq++.a*** és a ***libpq++.so*** fájl). Ez – kihasználva az objektumközpontú programozás fegyvertárát – egyesíti magában a PQ API finom részletezési lehetőségét és a PGEASY egyszerűbb programozhatóságát.

Egy SQL programozási környezetben alapkövetelménnyé vált, hogy egy gazdanyelvbe (esetünkben ez a C/C++ lesz) az SQL-parancsokat úgy ágyazhassuk be, mintha azokat egy SQL-konzolon adtuk volna ki. Ilyen például az `ORacle Pro` C vagy a PG negyedik típusú C/C++ API-készlete is (ennek helye a ***libecpg.a*** és a ***libecpg.so***). Itt az az alapelv, hogy egy ***program.pc*** nevű forrásprogramot készítsünk, amiben `EXEC SQL...` kezdetű makrókat helyezhetünk el, biztosítva az SQL-parancsok közvetlen beírását. A C/C++-változók és az SQL-parancsok közötti kapcsolattartást a változó neve elé elhelyezett kettős pont (:) karakter jelöli (ennek neve külső változó). Ebben a formában programunkat a C/C++-fordító nem tudja lefordítani, hiszen nem ismerheti az `EXEC SQL...` makrók feldolgozásának a módját. A PG rendelkezik egy `ecpg` nevű, beágyazott SQL-előfeldolgozóval, ami biztosítja a `*.pc` fájlok `*.c`-vØ alakítását, azaz egy olyan C/C++-program elkészítését, ami egy C/C++ fordítóprogrammal már lefordítható. A beágyazott SQL alapú program a natív API gyorsírásos változatának is tekinthető, hiszen az önműködően létrehozott C/C++-programot kézzel is megírhattuk volna.

Lényeges kiemelni, hogy az ismertetett négyféle API (sőt maga az adatbázis-kezelő kiszolgáló is) a Unix/Linux mellett Windows alatt is használható, így az operációs rendszerek különbsége PG-adatbázisunk elérésének nem emel gátat.

A továbbiakban nézzünk egy-egy példát az egyes API-k gyakorlati használatára! A programozási feladat minden esetben az lesz, hogy kilistázzuk a ***cs_adatok*** adatbázis-telefontábláját, majd kipróbálás gyanánt egy új sort (ez alól a `pgeasy` példa kivétel) szúrunk be a táblába.

4. lista PostreSQL-adatbázis natív módú elérése

```
#include <stdio.h>
#include <stdlib.h>

/* Ez az include fÆjl kell a nat v
   PostgreSQL elØrØshez */
#include "/usr/include/pgsql/libpq-fe.h"

int main(void)

char sqlStr[256]; // Itt lesz a mindenkori
                  // SQL-parancs.
PGconn *conn; // A kapcsolat le r ja.
PGresult *result; // Az eredmØnyhalmaz.
int i; // A for ciklushoz sz ksØges.

// Kapcsol dÆs az adatbÆzishoz, ami a
// gazdagØp 5432 kapujÆn vÆrakozik egy
// kapcsol dÆsi kØrØsre. Az adatbÆzisunk
// neve: cs_adatok
// user/password = postgres/inyiri
conn = PQsetdbLogin
➥("localhost","5432","","","cs_adatok",
➥"postgres","111111");

// MegvizsgÆljuk, hogy siker lt-e a
// kapcsol dÆs
if ( PQstatus(conn)==CONNECTION_BAD )

printf("!!! Hiba az adatbÆzishoz
➥kapcsol dÆskor...n");
exit(1);
...
```

A PQ (natív PG) API alapú mintapélda

A program teljes forráskódját az ***alap_pg.c*** fájl tartalmazza (4. lista, 44 CD Magazin/PostgreSQL könyvtár), amibe az egyes programsorok közé beírtam azokat a megjegyzéseket, amelyek elégségesek a működés megértéséhez.

Az ***alap_pg.c*** program fordítása és szerkesztése a következő paranccsal lehetséges:

```
# gcc alap_pg.c -o alap_pg.exe -lpq
```

A kapott ***alap_pg.exe*** program lefuttatása a következő eredményt jeleníti meg a képernyőn:

```
Nyiri Imre   --- 123456789
Kiss JÆnos   --- 123456789
PØntek TamÆs   --- 123456789
```

```
Janisovszky KÆroly  --- 11111
A postgreSQL pr baprogram V GE...
```

A PGEASY API alapú mintapélda

A program forráskódja az ***egyszeru_pg.c*** fájlban található (lásd *5. listánkon*). Vegyük észre, hogy ez a program nem kezeli explicit módon a munkamenetet, hiszen a kapcsolódás után az SQL-lekérdezésben a (`doQuery(...)`)-ben ez az adat nincs megadva.
Az ***egyszeru_pg.c*** program fordítása és szerkesztése a következő paranccsal lehetséges:

```
# gcc egyszeru_pg.c -o egyszeru_pg.exe -lpgeasy
```

Az ***egyszeru_pg.exe*** lefuttatása után a telefontábla tartalmát ez a program is kilistázza.

A beágyazott SQL alapú mintapélda

A program teljes forráskódját a ***beagyazott_pg.pc*** fájl (6. lista, 44. CD Magazin/PostgreSQL könyvtár) tartalmazza. Tekintettel arra, hogy a beágyazott SQL szabványos módszer, az ilyen programok készítését nagyon sok forrásból megismerhetjük. Nézzük tehát a „Listázás és egy beszúrás" mintaprogramot!
A futtatható program előállítása két lépésből áll. Az első lépésben a PG fejlesztői készletben található `ecpg` makrófeldolgozó segítségével a következő parancs révén előállítjuk a ***beagyazott_pg.c*** forrásfájlt:

```
# ecpg beagyazott_pg.pc
```

A kapott C-forrásprogramokat lefordíthatjuk

```
# gcc beagyazott_pg.c -o beagyazott_pg.exe
  ➥-lecpg -I/usr/include/pgsql
```

A ***beagyazott_pg.exe*** lefuttatása után az előző mintaprogramokhoz hasonló kimenetet kapjuk meg a képernyőn.

A C++ (PQ++) API alapú mintapélda

Ennek a mintapéldának a teljes forrása a ***cpp_pg.cpp*** fájlban (7. lista, 44. CD Magazin/PostgreSQL könyvtárában) található.
A program feladata ugyanaz, mint az eddigi C/C++ alapú példáknak. A `PGDatabase` osztály jelképezi a PG-adatbázis elérését és használatát. A forráskód egyszerűségét főleg az egységbezárás és az alapértelmezett értékek lehetőségének a kihasználása biztosítja.
A program fordítása és szerkesztése az alábbi paranccsal történhet, melynek eredményeképpen megkapjuk a ***cpp_pg.exe*** futtatható programot.

```
# gcc cpp_pg.cpp -o cpp_pg.exe -lpq++
  ➥-I/usr/include/pgsql
```

Ügyféloldali programozás Java-környezetben

A Java-környezet egyszerű, parancssoros használhatósága érdekében érdemes egy kis segítő héjprogramot használatba venni. Miért? A Java nyelvű forrásprogramok fordítása és futtatása is ugyanazt a dinamikus osztálymegkeresési (a fájlrendszerben az osztályok a ***.class*** fájlok) és -betöltési módszert követi. A dinamikusan betöltendő osztályok a `CLASSPATH` környezeti változóban megadott helyeken keresődnek. A héjprogram `SetCLASSPATH` függvénye önműködően beállítja a

5. lista PostreSQL-adatbázis easy módú elérése

```
#include <stdio.h>
#include <stdlib.h>

/* Ez az include fÆjl sz ksØges az easy
   PostgreSQL elØrØshez */
#include "/usr/include/pgsql/libpq-fe.h"
#include "/usr/include/pgsql/libpgeasy.h"

int main(void)

char sqlStr[256]; // Itt lesz a mindenkori
                  // SQL-parancs
char nev[50];
char szam[20];

// Kapcsol dÆs az adatbÆzishoz
connectdb("dbname=cs_adatok user=postgres
➥password=111111");

// VØgrehajtunk egy SQL select parancsot
// Figyelj k meg azt az eleganciÆt, ahogy
// a C/C++ nyelv seg tsØgØvel dinamikusan
// eliÆll tjuk az SQL-parancsot!
sprintf(sqlStr, "select %s, %s from
➥telefon", "nev", "szam");
doquery( sqlStr );

// Ki rjuk az E tÆblÆt a konzolra
// A fetch(...) f ggvØny vÆltoz  szÆmœ
// ØrtØket tartalmazhat. Most a nev Øs
// szam nevß meziket kØrj k le.
while ( fetch(nev,szam) != END_OF_TUPLES )

printf("N?v: %s Sz?m: %sn", nev, szam);


// Az adatbÆzis-kapcsolat bontÆsa
disconnectdb();

printf("A PostgreSQL pr baprogramnak
➥V GE...n");
return 0;
...
```

`CLASSPATH`-t. Ennek csak annyi az előfeltétele, hogy a használandó ****.jar*** és ****.zip*** fájlok hivatkozásai a parancsfájlban megjelölt ***ext*** könyvtárban létre legyenek hozva.
A SuSE 8.0 alatt a ***/usr/share/pgsql*** könyvtárban található meg a ***jdbc7.1-1.2.jar*** fájl, ami a PG JDBC-meghajtója (megjegyzés: ez a jar fájl az Internetről is letölthető, sőt egy 1.3-as változata is létezik már). Erre a fájlra hozzunk létre egy hivatkozást az ***ext*** könyvtárban. A hivatkozás neve bármi lehet, de ****.jar*** legyen a fájl kiterjesztése. A segítő parancsfájl a ***ffjava.sh*** fájlban (8. lista, 44. CD Magazin/PostgreSQL könyvárában) található.
Az ***ffjava.sh*** parancsfájl fordításra és futtatásra is használható. A ***pg_teszt.java*** program fordítása a ***fjava.sh -c pg_teszt.java***, míg a lefordított program `pg_teszt` osztályának elindítása az `ffjava.sh -r pg_teszt` paranccsal lehetséges.

6. lista PostreSQL-adatbázis beágyazott módú elérése

```
EXEC SQL WHENEVER SQLERROR sqlprint;

int main(void)

//
// Az itt megadott vÆltoz k hasznÆlhat k az
// EXEC SQL-kifejezØsekben k lsı
// vÆltoz kØnt. Erre val  hivatkozÆskor
// ":" kettıs pontot tesz nk elØ.
//
EXEC SQL BEGIN DECLARE SECTION;
char nev[50];
char szam[20];
char sqlStr[256];
EXEC SQL END DECLARE SECTION;

// Kapcsol dÆs az adatbÆzisunkhoz.
EXEC SQL CONNECT TO
cs_adatok@localhost:5432 ➥user postgres
using "111111";

// Ez lesz az SQL-parancs
sprintf(sqlStr, "select %s, %s from
➥telefon", "nev", "szam" );

// Az SQL-parancs elıfeldolgozÆsa
// (optimalizÆlÆsi lehetısØg)
EXEC SQL PREPARE s_telefon FROM :sqlStr;

...
```

Ennyi előkészület és környezetkialakítás után írjuk meg szokásos 1 select 1 insert típusú próbaprogramunkat Java nyelven is! Ennek a nyelvi környezetnek a nagy előnye az, hogy szinte minden módszerre, így az adatbázisok kezelésére is átgondolt objektumközpontú, szabványos eszközrendszert bocsát a rendelkezésünkre.
A program szellemisége egyszerű: létrehoz egy `Connection` objektumot (melynek `conn` a neve), ami egy adatbázis-kapcsolat objektumközpontú megfogalmazása. A `conn` objektum képes egy SQL-parancsot képviselő `Statement` osztálybeli objektumot szolgáltatni, ami már egy-egy valóságos SQL-parancs kiadását is biztosítja. A `select` parancsok eredménytábláját egy `ResultSet` osztálybeli `rs` objektum képviseli. Érdemes észrevenni, hogy itt sincs semmi új a nap alatt, hiszen szemléletében szinte az összes objektumközpontú adatbázis-kezelő ezt az osztály-együttműködési módszert követi.
A programban (fájl neve ***pg_teszt.java***) elhelyeztem azokat a megjegyzéseket, amik a kód könnyebb megértését szolgálják (9. lista, 44. CD Magazin/PostgreSQL).
A programnak létezik egy `getMagyarString()` tagfüggvénye, ami példát mutat arra, hogy az `rs.getString()` mellett más módon is hozzájuthatunk az oszlopok adataihoz. Erre a módszerre azért volt szükség, mert egy PG-adatbázis alapértelmezett kódolása az SQL_ASCII, ugyanakkor a karakterláncokat a Java belül Unicode-ban tárolja. Itt az volt a gond, hogy egy C++-programmal ISO-8859-2-ben dolgoztunk, aminek `insert`, `update` adatait a PG-adatbázis a saját

7. lista PostreSQL-adatbázis C++-elérése

```
...

int main(void)

char sqlStr[256]; // Itt lesz a mindenkori
                  // SQL-parancs
int i; // A for ciklusvÆltoz ja

// AB az adatbÆzis-objektum. A lØtrehoz
// f ggvØnyben az adatbÆzishoz kapcsol dÆs
// is megt rtØnik
PgDatabase AB("dbname=cs_adatok
➥user=postgres password=111111");

// Ha nem sikeres a connect, akkor vØge
if ( AB.ConnectionBad() )

cerr << "!!! Hiba az adatbÆzishoz
➥kapcsol dÆskor...n";
exit(1);

...
```

SQL_ASCII kódolásában bájthűen tárolta. Ezt azonban (az ő, ű betűket) a Java `rs.getString("nev")` tagfüggvény nem helyesen olvasta be. A végső megoldás az lett, hogy az adatbázis kódolása LATIN2-re lett meghatározva. Ekkor már az egyszerű `getString()`, `setString()` (lásd a `beszur()` forrását) tagfüggvények hibátlanul működnek.
A teljesség kedvéért nézzük meg azt a lehetőséget is, hogy miként történik egy tárolt eljárás meghívása a Java nyelvből. A jdbc leírás szerint ezt a feladatot a `CallableStatement interface`-t megvalósító osztály tudná végrehajtani, ez a felület azonban a PG-ben nincs megvalósítva. Helyette ezt a feladatot egy `select` eljárásba ágyazott függvényhívással oldhatjuk meg. A példaprogram a már létrehozott `MyFunc(integer)` meghívására mutat egy esetet (forrásprogram ***pg_storproc.java***). A PG-ben a `select f ggvØny()` forma is használható lenne, azaz a dualtáblára nincs is szükségünk, ennek kihasználása viszont a program hordozhatóságát rontaná (lásd a 10. listát, ami a 44. CD Magazin/PostgreSQL könyvtárában található).
A PG programozói felületek a `select`, `insert`, `update`, `delete` (DML=Data Manipulation Langauge műveletek) parancsokon kívül a többi SQL-utasítás (DDL=Data Definition Language műveletek) kiadását is támogatják. Erre rövid példa a következő három sor, ami az adatbázisból kitörölné a telefontáblát:

```
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery("DROP TABLE
telefon");
st.close();
```

Grafikus segédprogramok

A `psql` nagyszerű eszköz adatbázisaink objektumainak kezelésére, de sokszor egy-egy grafikus eszköz használata is nagyon jól jön. A phpPgAdmin PHP-ben megírt webes ügyfél. A phpPgAdmin telepítése egyszerű. A webkiszolgálónk dokumentumterületére egyszerűen másoljuk be az összes fájlt, amit a `phpPgAdmin.tar.gz` tartalmaz. Én az Apache HTTPD-t

használom, dokumentumgyökerem: */var/www/htdocs*. A fenti tar.gz fájl-t e könyvtár alatt egy phpPgAdmin könyvtárba csomagoltam ki, így a böngészőből a ➲ http://gépnév/phpPgAdmin/index.php címmel el tudom indítani. Ezzel az eszközzel a leggyakoribb feladatokat el lehet végezni. Könnyen menedzselhetjük vele adatbázis-objektumainkat. A kioldók és tárolt eljárások készítése is eléggé kényelmesen elvégezhető vele. Nagy erénye, hogy bárhonnan grafikusan is elérhetjük az adatbázisunkat, hiszen csak egy böngésző szükséges hozzá az ügyféloldalon. A phpPgAdmin honlapja és letöltési helye a ➲ http://phppgadmin.sourceforge.net címen található.
A másik elterjedt GUI-ügyfél a `pgaccess`. Ezt az eszközt mostanában ismét feszített iramban elkezdték továbbfejleszteni, ezért javaslom, hogy folyamatosan kövessük az új változatokat, amelyek remélhetőleg elsősorban az űrlap- és riportkészítési lehetőségekben fognak újdonságokat hozni.
Egyszerű beviteli űrlapok és riportok készítésére már most is van lehetőség. A pgaccesshez egy HTML alapú leírás és egy oktatóanyag tartozik, így aki erről a programról többet szeretne tudni, olvassa el ezeket az írásokat. A pgaccesss a ➲ http://www.pgacces.org webcímen érhető el.
Befejezésül fontos kiemelni, hogy a Java-felületen sokféle grafikus adatbázis-kezelő GUI-eszköz létezik. Ezeknél általában két jellemzőt kell beállítani:

- Az adatbázis `JDBC driver` osztályának a nevét és a helyét (org.postgresql.Driver).
- A kapcsolódási karakterláncot (jdbc:postgresql://localhost:5432/cs_adatok).

A kapcsolók megadása után már használhatjuk is legújabb Java swinges adatbázis-kezelő felületünket.

Windows alapú ügyfélprogramok készítése

A PG natív elérését lehetővé tévő DLL-könyvtárak Windowson is rendelkezésünkre állnak, néha azonban egyszerűbb az ODBC-meghajtót használni. A PG ODBC-meghajtó Windows telepítőcsomag formájában letölthető az Internetről.
A PG ODBC-meghajtó telepítését követően PG adatbázisokra mutató ODBC adatforrásokat hozhatunk létre. Az adatforrásra az Excel, MS Access... programokból ***odbc_cs_adatok*** néven hivatkozhatunk. A lényeg az, hogy ezen alias név mögött a gazdagép-, a kapu- és az adatbázisnév-adatokat tároljuk.
A C/C++-, Java-ügyfélprogramok megértése után valószínűleg már nem is csodálkozunk azon, hogy miért pont ezek az adatok szükségesek egy TCP/IP-n keresztül történő PG-adatbázis ODBC-n keresztüli eléréséhez.
Érdekes lehetőséget kínál Delphi-, Borland C++ Builder-programozók számára az az út, hogy egy ODBC-kapcsolatot egyszerűen BDE (Borland Database Engine) alias névre lehet alkalmazni, azaz az ***odbc_cs_adatok*** névre hivatkozva a BDE Administrator program segítségével meg lehet határozni egy ***bde_cs_adatok*** adatbázis alias nevet, amit a Delphi-programokban már közvetlenül használhatunk. Nyitva áll tehát az út a különféle windowsos ügyfélprogramok készítése előtt.

Egyéb lehetőségek

A PG-adatbázisok természetesen még sok népszerű programozási eszközből kezelhetők. Ezek közül a legfontosabb talán a PHP, amiben olyan kiváló dolgok készültek, mint a már említett phpPgAdmin webes PG-adatbázis ügyfélkörnyezet. A PHP és a PG kapcsolatáról a Linuxvilág 2000. decemberi (I. évfolyam, 2. szám) és 2001. június–július havi (II. évfolyam, 6–7. szám) számai tartalmaznak egy-egy érdekes cikket. A PHP PG programozási lehetőségeit egyébként az Internetről letölthető és nagyrészt már magyarra is lefordított, több mint 1000 oldalas PHP-kézikönyv is részletesen tartalmazza.
A Perl nyelvhez is létezik egy PG-csomag, amit SuSE 8.0 alatt a postgresql-perl csomag telepítésével kapunk meg. A Perl az adatbázis-kezelő függvényeket egy DBI modulban (Database Interface Modul) tárolja, de minden adatbázis-kezelőhöz még a megfelelő adatbázis-kezelőhöz tartozó DBD (DataBase Dependent) modult is telepíteni kell. Nézzük meg szokásos mintalekérdezésünk Perlben megírt változatát!

```
#!/usr/bin/perl
use Pg; # HasznÆljuk a Pg csomagot
$conn = Pg::connectdb("dbname=cs_adatok");
# Kapcsol dÆs
die $conn->errorMessage unless
          ➥PGRES_CONNECTION_OK eq
          ➥$conn->status; #HibakezelØs
# A szokÆsos select parancsunk kiadÆsa j n:
$result = $conn->exec("select nev, szam from
                ➥telefon");
#HibakezelØs
die $conn->errorMessage unless PGRES_TUPLES_OK
          ➥eq $result->resultStatus;

// Ki rjuk a konzolra a lekØrdezØs eredmØnyØt:
while (@row = $result->fetchrow) {
  print @row, "\n";
}
```

A Linuxvilág 2002. augusztusi száma (III. évfolyam, 8. szám) egy érdekes lehetőséget, a clip (Clipper) fejlesztői környezetet mutatja be, ezen belül arra is választ ad, hogy hogyan lehet PG-adatbázisokat Clipper nyelvű programokból elérni.
A fentieken kívül a PG programozhatóságát TCL/Tk (a tárgyalt pgaccess GUI-ügyfél is ebben készült) és Python-környezetben is megvalósították.
Remélem, hogy a röviden bemutatott PG programozási eszközök gazdagsága mindenkinek felkeltette az érdeklődését a PostgreSQL adatbázis-kezelő iránt, amit nyugodtan tekinthetünk a világ egyik legkiforrottabb adatbázis-kezelő rendszerének.
Kellemes programozást kívánok mindenkinek!

Irodalomjegyzék

- A PostgreSQL alapleírása (a Linux-terjesztések is tartalmazzák): ➲ http://www.hu.postgresql.org (a PG honlapjának magyarországi tükörkiszolgálója)
- ***Bruce Momjian***: PostgreSQL Introduction and Concepts (Addison-Wesley, 2001). E könyv pdf formátumú alakja szabadon letölthető az Internetről.
- ***Jeffrey D. Ullman–Jennifer Widom***: Adatbázisrendszerek (Alapvetés) (Panem, 1998)
- ***Joe Celko***: SQL felsőfokon (Kiskapu Kft., 2002): ➲ http://www.kiskapu.hu, ISBN 963 9301 20 5

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

Betűvetés

Gyakran tapasztalhattuk, hogy a Linux grafikus felületén (főleg a Netscape-pel megjelenített weboldalak esetében) a betűk gyakran nagyon csúnyák, kicsik, mi több: olvashatatlanok.

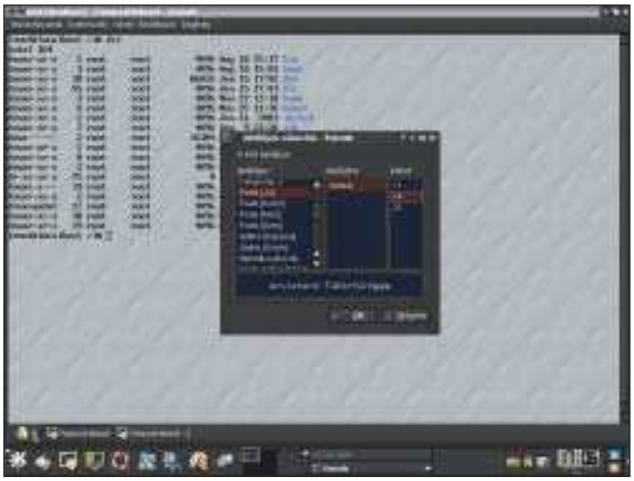
Enehézségre a megoldást nem csak a monitor közelebb húzása, vagy egy szemüveg feliratása jelentheti. A fenti „csúfságok" elkerülése érdekében telepíthetünk például pár TrueType betűtípust.
A leggyakoribb gond a következő: a Netscape-pel folytatott világhálón keresztüli vad száguldozás során egyes weboldalak szövege olyan kicsi és rút betűtípussal jelenik meg, hogy elolvasásuk komoly megterhelést jelent látószerveinkre nézve. Windows alól azonban ehhez hasonló jelenséggel még a legritkább esetben sem találkozhattunk. Vajon minek „köszönhető" ez a kétségbeejtő helyzet?
Ez nem a Netscape hibája és nem is a grafikus rendszeré.
A galibát egyszerűen az okozza, hogy a terjesztések többségében általában csak néhány előre megrajzolt állandó méretű (10, 12, 14, 18 és 24 pontos) betűtípus található meg. Gondok akkor mutatkoznak, ha a megjelenítendő HTML-állományt a cseles webtervező teletűzdeli mindenféle <FONT> tagokkal, és az alapbeállítástól eltérő méretben kéri a megjelenítést a böngészőtől, amelynek eredményeképp a betűtípusok már-már mikroszkopikus méreteket öltenek az ablakban.
Ha szép és jól olvasható betűkészletet szeretnénk X-ünk alá, érdemes megfontolni néhány TrueType betűtípus telepítését. A továbbiakban erről lesz szó.

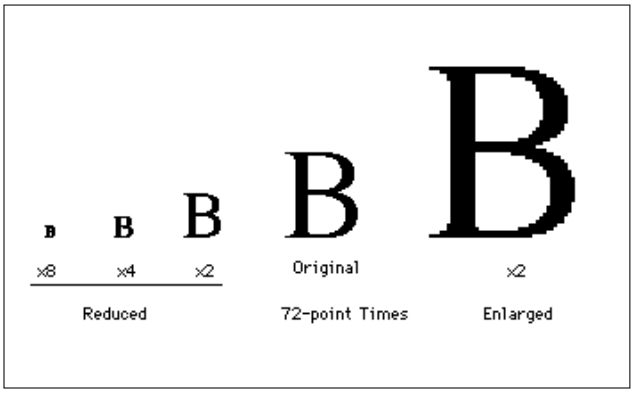
Betűtípusok

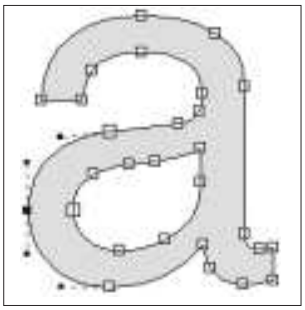
Még mielőtt bármibe is belekezdenénk, nem árt, ha ejtünk néhány szót a legelterjedtebb betűtípusokról. Elsőként a fixpontos vagy bitkép betűtípusokat érdemes megemlítenünk. Különböző jogi okok miatt a legtöbb terjesztés többnyire ilyen betűtípusokkal van felszerelve. Legfontosabb tulajdonságuk, hogy a betűk kinézete pontról pontra van tárolva egy mátrixban, tehát kirajzolásuk viszonylag gyorsan elvégezhető (és a művelet sokkal kevesebb memóriát igényel, mint a TrueType, illetve Type1 betűtípusok megjelenítése). Másik érdekes tulajdonságuk, hogy az ilyen betűtípusokkal megjelenített szöveg minden rendszeren (illetve bármilyen nyomtatóval kinyomtatva) ugyanakkora méretű lesz. A fixpontos betűtípusokkal leggyakrabban a szövegszerkesztőknél, illetve a terminálablakoknál találkozhatunk, de ilyet használ maga a konzol is.
A fixpontos betűtípusok két formátumát támogatja az X: a ***.pcf***-et és a ***.bdf***-et. A kettő között az az érdembeli különbség, hogy míg az első bináris, addig a másik szöveges állomány, tehát az utóbbi módosításához egy egyszerű szövegszerkesztő is elegendő. (Megjegyezzük, hogy lehetőség nyílik a windows-os ***.fon*** állományok ***.pcf***-é való átalakítására. Ha valaki netán ilyesféle életcélokat tűz ki maga elé, a megvalósításhoz csupán a Wine-ban – ez egy linuxos Windows-emulátor – megtalálható átalakítóra van szüksége.)
A betűtípusok másik nagy családját az úgynevezett méretezhető betűtípusok alkotják, ezeknek két elterjedt fajtájuk van: az Adobe által kifejlesztett Type1 és a Windows világában rendkívül elterjedt TrueType.
Itt a fixpontos betűtípusokkal ellentétben az egyes karaktereket alkotó képpontok helyét nem mátrixokban tárolják, hanem vektorosan, úgynevezett Bézier-görbékben (Bezier paths).
A Bézier-görbéket az ívek matematikai leírására használják, ahol az adott görbét az ív csomópontjai és a hozzájuk tartozó érintők határozzák meg. A számítástechnikában számtalan helyen találkozhatunk velük, például grafikus alkalmazásoknál, illetve ez az alapja a PostScript nyelvnek is.
A Type1-et kifejezetten a PostScript nyelvhez találták ki, azaz a papírra történő nyomtatáshoz. Unixos körökben széles körű támogatást élvez, így az X-szel is „jól kijön".
A TrueType-ot az Apple fejlesztette ki, és a Microsoft világában rendkívül népszerű. Sok közös vonás fedezhető fel a TrueType és a Type1 betűtípusok között. Számunkra a legfontosabb különbség közöttük az, hogy a Type1-eket nyomtatásra tervezték, ezért a grafikus felületek 75-100 DPI-s felbontásán borzalmasan mutatnak. A monitoron tehát a TrueType betűtípusok sokkal jobban festenek.

TrueType betűtípusok beszerzése

Írásunkban egy teljes receptet fogunk bemutatni, amelynek segítségével grafikus rendszerünkhöz könnyedén telepíthetünk mindenféle TrueType betűtípust.
Az első és legfontosabb kérdés: honnan szerezzük be a TrueType betűtípusokat? Mint már említettük, különböző jogi megfontolások miatt a Linux-terjesztések többsége nincsen TrueType betűtípusokkal kibélelve, általában csak bitképesre, illetve néhány Type1-esre lelhetünk. Habár a mai újabb terjesztések esetében javult egy kicsit a helyzet, nem valószínű, hogy minden igényt kielégítenek. (Az igazat megvallva elég rossz tapasztalataim gyűltek össze az alapból jelenlévő TrueType betűtípusokkal. Például előfordult már olyan eset, hogy egyes átalakító alkalmazásoknak „megfekszik a gyomrát" vagy karakterek hiányoznak a betűkészletből. De azt azért elmondhatjuk, hogy az idő előrehaladtával és a terjesztések változatszámainak növekedésével egyre jobbak a kilátások.)
Térjünk vissza a TrueType betűtípusok beszerzésének a kérdésére! Akadnak olyan elvetemült emberek, akik egyenesen a Windowsból lopják őket. Ha van kéznél törvényesen használt Windowsunk, megtehetjük, ha nem, akkor is, de ezzel megsértjük a hatályban lévő jogszabályokat. Ezért azt javasoljuk, hogy nézzünk körbe az Interneten, amely telis-tele van ingyenesen letölthető és szabadon használható betűtípusokkal. Régebben a Microsoft weboldalán rengeteg jól használható betűtípusokra bukkanhattunk (➲ http://www.microsoft.com/truetype/fontpack/win.htm), amelyet az Internet Explorer lelkes felhasználói számára tartottak fent. Ki tudja, miért – talán mert zokon vették azt, hogy a linuxos tábor is hasznot húzott belőle –, most már nem elérhetőek. Mindenesetre az oldalon azért bőséges







hivatkozásgyűjteményt találhatunk. Debian alatt létezik egy csomag, az msttcorefonts, ami a `wget`-et használva letölti a hálóról a Microsoft szabad készletét (Andale, Arial Black, Arial, Verdana és a többi), majd telepíti. Más betűkészletekkel kézzel kell elbánnunk.

A telepítés

A beszerzett betűtípusok beüzemeltetéséhez szükségünk lesz egy freetype nevű csomagra, amely szinte minden terjesztésben megtalálható. A freetype tulajdonképpen egy könyvtárcsomag, amely a TrueType betűtípusokat gyorsan és szépen meg tudja jeleníteni. Tartalmaz még egy `ttf2bdf` nevű segédprogramot is, amellyel kedvenc TrueType betűtípusainkat fixpontosokká alakíthatjuk át az általunk meghatározott méretben. Így ha nincs kedvünk, nem muszáj a méretezhető betűtípusok telepítésével vesződnünk. Szerencsére ez a legtöbb terjesztésben megtalálható. Első lépésként hozzunk létre egy könyvtárat, és másoljuk oda az összes telepíteni kívánt betűtípust. Ha ezzel kész vagyunk, adjuk ki az `ttmkfdir >fonts.dir` parancsot – ez tulajdonképpen nem tesz mást, mint listát készít a könyvtárban található betűtípusokról.

Akik 4-es változatú X-et használnak (a frissebb terjesztések már mind ezt teszik), azok szerencsés helyzetben vannak, ugyanis ez a rendszer már mind TrueType-, mind Unicode-támogatást tartalmaz. A feladatunk tehát már csupán annyi, hogy az X kiszolgáló beállítási állományában (amely általában a */etc/X11/* könyvtár alatt található) megkeressük a ***Section/Files*** bejegyzést. Itt mindenképp találnunk kell egy ***FontPath*** kezdetű sort. Ezt hagyjuk békén, de alá írjuk be a következő sort:

```
FontPath "/ahova mentett k/ttffontok"
```

A FontPath meghatározás helyébe természetesen annak a könyvtárnak a nevét kell írnunk, ahová a TrueType fontokat helyeztük. A művelet befejezéseképpen indítsuk újra a grafikus rendszert.

Előfordulhat, hogy miután az itt felsorolt beállításokat elvégeztük, az X nem indul el. Ilyenkor ne essünk pánikba, egyszerűen arról van szó, hogy a beállítási állomány módosításakor valamit elállítottunk vagy rosszul írtunk be. Ezért azt javasoljuk, hogy még mielőtt buzgón nekilátnánk a FontPath állítgatásának, készítsünk biztonsági másolatot az eredeti beállításokról.

Aki még a régebbi, például a 3-as X-et használja, annak sajnos meg kell ismerkednie egy betűtípus-kiszolgáló nevű jelenséggel. Ebben az esetben nem az X kiszolgáló belső eljárásai foglalkoznak a betűtípusok kezelésével, hanem kapcsolatba lép a betűtípus-kiszolgálóval, és vele végezteti el a velük kapcsolatos feladatokat. Ilyenkor az X beállítási állományában a FontPath meghatározásnál egy `unix/:`*kapuszÆm*-ot vagy `unix/:-1`-et találhatunk.

A beállításban annyi lesz a különbség az előzőekhez képest, hogy az új betűtípusokat nem az X kiszolgálóhoz „adjuk hozzá”, hanem a betűtípus-kiszolgálóhoz. Manapság az `xfs` nevű betűtípus-kiszolgáló a legelterjedtebb, amelynek egyik legnagyobb erőssége az, hogy a magyar ékezetes betűtípusokat is támogatja. Ezért mi is az `xfs`-t vesszük alapul.

Ez sem túlzottan ördöngös feladat. Az `xfs` beállítási állományát `/etc/X11/fs/config` néven találjuk. Itt keressük meg a `catalouge =` kezdetű sort, ahol egymás után fel vannak sorolva a betűtípusokat tartalmazó könyvtárak, vesszővel elválasztva egymástól. Írjuk hozzá azt a könyvtárat, ahová az új betűtípusokat pakoltuk, majd indítsuk újra az `xfs`-t:`/etc/rc.d/init.d/xfs restart`. Figyelem! Ha az X kiszolgáló a betűtípus-kiszolgálót használja, ügyeljünk rá, hogy az X-et mindig csak a betűtípus-kiszolgáló után indítsuk el. Ellenkező esetben a grafikus rendszer indulási kísérlete csúnyán kudarcot vall.

A frissen telepített betűtípusokat az `xfontsel` program segítségével meg is tekinthetjük. Most már a Netscape ***Preferences/Edit/Fonts*** menüjében bármilyen kódoláshoz bármilyen nekünk tetsző méretet beállíthatunk.

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.

A jó időzítés titkai

Amikor egy felhasználót arra kérünk, hogy soroljon fel minél több olyan eszközt, amellyel bevitel–kivitel hajtható végre, az esetek nagy százalékában biztosan el fog feledkezni a számítógép órájáról.

Első hallásra valóban furán hangzik, hogy az órát is B-K eszköznek kell tekintenünk, pedig látni fogjuk: valóban az, és nem is akármilyen! Nélkülözhetetlen feladatokat bíz rá ugyanis az operációs rendszer.

Az első kérdés, amit érdemes lenne megvizsgálni, hogy tulajdonképpen mit is értünk a számítógép órája alatt. Az elnevezés megtévesztő lehet, ugyanis a számítógépekben található órák egyáltalán nem hasonlítanak az éjjeliszekrényünkön található vekkerünkhöz, de még a Rolexünkkel sincsenek semmiféle rokonságban. Ezek az órák ugyanis olyan eszközök, amelyeknek az a feladatuk, hogy bizonyos időközönként megszakítást hajtsanak végre.

A felhasználó magával az órával általában csak akkor találkozik (és akkor is csak közvetett módon), amikor lepillant a képernyő sarkába, ahol ablakkezelőnk lelkesen mutatja a pontos időt.

Az operációs rendszer számára azonban elengedhetetlenebb szolgáltatásokat nyújt. Vegyük például azt az esetet, amikor egy folyamat semmilyen beviteli–kiviteli műveletet nem hajt végre, csak számolgat magában a végtelenségig. Ilyenkor az ütemezőnek esélye sem lenne megakadályozni, hogy az ilyen folyamatok teljes egészében kisajátítsák maguknak a processzort (ugyanis B-K művelet hiányában nem keletkezik megszakítás, lásd a Linuxvilág 2002. augusztusi számát, 48–51. oldal).

Ehhez hasonló esetekkor válnak hasznossá az óra által létrehozott megszakítások (amelyek előbb vagy utóbb biztosan bekövetkeznek, függetlenül attól, hogy van-e valamilyen B-K művelet folyamatban), így az ütemezőnek lehetősége nyílik felfüggeszteni az adott folyamat futását. Az órát még sok minden másra is felhasználja az operációs rendszer, hogy mire, arra még később visszatérünk.

Érdekes, hogy az órát is a beviteli-kiviteli eszközök közé soroljuk, miközben semmiben sem hasonlít például a lemezekhez vagy a terminálokhoz. Sőt még arról sem beszélhetünk, hogy az óra karakteres vagy blokkeszköz-e. Ám egy másik nézőpontból tekintve egyértelműen B-K eszköz. A felhasználó szemszögéből nézve az óra nem a hétköznapi értelemben vett B-K eszköz, mivel nem látja, nem foghatja kézbe, hiszen egybe van építve az alaplappal. Most gondoljuk át az egészet még egyszer! Mit csinál az óra? Megszakításokat hoz létre. Akkor szükség lesz egy olyan eljárásra, amelyik válaszol ezekre a megszakításokra, azaz egy órakezelőre. Mivel ez is megszakításokat kezel, mint például a lemezkezelők, neki az összes többi B-K eszközkezelővel egy szinten kell futnia. Akkor tehát az operációs rendszer szempontjából az óra is egyszerű B-K eszköz. Igaz, se nem blokkos, se nem karakteres, mégis ugyanúgy kell vele foglalkozni, mint a többi egységgel.

A számítógépekben általában kétféle órát alkalmaznak. Az első közvetlenül a táphoz kapcsolódik, és minden egyes fázisváltáskor megszakítást okoz (ez viszonylag gyakori esemény, egy másodpercben körülbelül ötvenszer-hatvanszor következik be). A másik órában egy kristály oszcillátor van, ami nem más, mint egy megfelelő formára hozott és nyomás alá helyezett kvarckristály. Ennek köszönhetően kihasználhatjuk a kvarckristály jellétrehozó tulajdonságait, ezek ugyanis nagyon megbízhatóak. Az órához tartozik egy számláló is, és minden, a kvarckristály által keltett jelre csökkenti ennek a számlálónak az értékét. Amikor a számláló eléri a nullát, létrejön a megszakítás. A két óra között az érdembeli különbség az, hogy az elsőnél nem határozhatjuk meg a megszakítás rezgésszámát (frekvenciáját, azaz az órát nem programozhatjuk), míg ennél megmondhatjuk, hogy a megszakítások milyen időközönként történjenek. Az ilyen órák másik jellemző tulajdonsága, hogy általában több egymástól független programozható órát tartalmaznak.

Ezenkívül találhatunk még egy órát, amire akkor van szükség, amikor a gépet kikapcsoljuk (ezt gyakran háttérórának is nevezzük). Ez gondoskodik ugyanis arról, hogy a pillanatnyi dátum és idő ne felejtődjék el. Ezek a karórákhoz hasonlóan elemmel működnek és nagyon keveset fogyasztanak. (Ma már azonban egyre divatosabbá válik, hogy a pontos időt is az Internetről „töltsük le”. Egy különleges protokoll segítségével megoldható két számítógép órájának az összehangolása.)

A legtöbb operációs rendszer tehát az órát is beviteli–kiviteli eszközként tartja számon, így az órakezelő is a többi eszközmeghajtóval együtt fut. Maga az eszközkezelés rendkívül egyszerű, csupán a létrejövő megszakításokat kell kezelni.

Az órakezelőnek azonban rengeteg más feladata is van, nézzük, melyek ezek!

A pontos idő

Kezdjük a legegyszerűbbel, a pontos idő (vagy valós idő) karbantartásával, ami tulajdonképpen nem jelent mást, mint azt, hogy számoljuk az óramegszakításokat. A Unix-rendszerek a pontos időt úgy tárolják, hogy erre a célra egy számlálót tartanak fenn, és abban mindig egy meghatározott alapidőpont (általában 1970. január 1-je, déli 12 óra) óta eltelt órajelek vagy másodpercek számát tárolják.

Ez azonban felvet egy bosszantó kérdést, ha ugyanis az eltelt órajeleket számoljuk, könnyen bajba kerülhetünk, ha a számláló méretének a megválasztásakor nem vagyunk eléggé körültekintőek. Vegyünk példának egy 32-bites számlálót, amely első hallásra nagyon sok órajel ábrázolására képes (több mint négymilliárdra), de ha utánaszámolunk, rájövünk, hogy kevesebb mint két év alatt túlcsordul.

A legkézenfekvőbb megoldásnak az tűnik, ha az órajelek számlálására nem 32-bites, hanem 64-bites számlálót használunk. Ezzel csak az a gond, hogy egy másodperc alatt sokszor meg kell változtatnunk a számláló értékét, ami rendkívül időigényes. Ezért kedvezőbb az a megközelítés, ha a számláló 32 bit marad, és az órajelek helyett az eltelt másodpercek számát raktározzuk. Ebben az esetben szükség van egy másik számlálóra, ahová a beérkező órajeleket tesszük. Egy 60 Hz-es óra esetében egy másodperc alatt pontosan 60 megszakítás kelet-

kezik, így amikor az órajelszámláló eléri a hatvanat, lenullázzuk, és a másodpercszámláló értékét eggyel növeljük. Sajnos ezzel a módszerrel is gond van: csak 136 évig alkalmazhatjuk, utána a másodperceket számláló 32-bites változó túlcsordul. Ennek következtében egy harmadik megoldásra is szükség van, ami kerüli a nagy méretű számlálók használatát, de nincs benne ez a 136 éves korlát. A megoldás nagyon egyszerű: nem egy meghatározott alapidőponttól (például 1970-től) kezdve számolunk, hanem a rendszer indulása óta eltelt órajeleket figyeljük. Amikor a rendszer betöltődik, kiolvassa a háttéróra állását, átváltja az alapidőponttól eltelt másodpercekre, majd raktározza a memóriában. Ha valaki a pontos időre kíváncsi, csupán egy egyszerű összeadás műveletre lesz szükség: a rendszer indulásának időpontját össze kell adni a számlálóval.

Korábban már említettük, hogy ha a gépünkben nem lenne óra, az operációs rendszernek esélye sem lenne felügyelni az olyan folyamatokat, amelyek futás közben egyetlen megszakítást sem okoznak (azaz egyetlen B-K műveletet sem hajtanak végre, csak számolgatnak magukban csendesen). Ez azonban még csak a jéghegy csúcsa, ezek a folyamatok ugyanis egyszer várhatóan abbahagyják a futást, esetleg mégis elvégeznek egy B-K műveletet. De ha egy folyamat például végtelen ciklusba kerül, az idők végezetéig várhatunk.

Szerencsére ilyen esetek nem fordulhatnak elő, mivel minden számítógép rendelkezik órával, így bármi történjék is, az ütemező előbb vagy utóbb biztosan „szóhoz fog jutni”, azaz lehetősége lesz megakadályozni, hogy akár egy B-K műveleteket kerülő folyamat is sokáig fusson. Ezt nyilván úgy éri el, hogy az éppen futó folyamathoz egy számlálót rendel, amelybe beírja, hogy a folyamat hány órajelen keresztül futhat. A számláló értékét minden órajel-megszakításkor csökkenteni kell, és ha eléri a 0-t, akkor az éppen futó folyamatot fel kell függeszteni, és a futási lehetőséget egy másiknak át kell adni.

A gyakorlatban nagyon fontos kérdés, hogy az operációs rendszer melyik része végezze például a valós idő, illetve az éppen futó folyamathoz rendelt számláló karbantartását. Mert például az utóbbi ugyan az ütemező feladata lenne, de bizonyos szempontból logikusabb, ha az órakezelő végzi, és ha letelt az egy folyamatra szánt idő, majd ő hívja meg az ütemezőt. Ezek olyan tervezési kérdések, amelyek felvetésekor sokféle szempontot figyelembe kell venni.

E szempontok közül talán az egyik legmeghatározóbb a sebesség kérdése. Az órajel-megszakítás ugyanis nagyon gyakori jelenség, másodpercenként sok ilyen történik. Ezért mindig arra kell törekedni, hogy az órakezelő kódja minél gyorsabb legyen. Az órakezelő feladatait nagyjából két részre oszthatjuk. Az első csoportot azok a műveletek alkotják, amelyeket kivétel nélkül minden órajel-megszakításkor el kell végezni, a másikat pedig azok, amelyeket csak bizonyos esetben. Így nem rossz ötlet ezeket a feladatokat különválasztani, és más-más szinteken végrehajtani.

Hogy egy kicsit jobban értsük, miről van szó, nézzünk egy példát ennek a megvalósítására. Példánk alanya most a Minix nevű operációs rendszer lesz, amelyből maga a Linux is sokat tanult. Mivel a Minix jellemzően folyamatokra épülő rendszer, minden folyamat (és eszközkezelő) üzenetküldéses módszer segítségével tartja a kapcsolatot egymással. Ha egy bizonyos eszköztől beérkezik egy megszakítás, a megszakításkezelő üzenetet küld a megfelelő (éppen blokkolt állapotban lévő) eszközkezelőnek, ami az üzenet hatására felébred, és nekilát az esemény kezelésének. Ám ez az óra esetében egyáltalán nem hatékony, mivel az üzenetküldés viszonylag lassú módszer. Megoldást jelenthetett volna, ha ebben az esetben felhagynak a folyamatstrukturáltság elveivel, és egyszerűen egy közvetlen rutinhívás segítségével a vezérlést egyből az órakezelőnek adják át. A Minixet azonban nem „éles” rendszernek tervezték, hanem oktatási célokra, így működését mindenki szabadon tanulmányozhatja. (Ennek köszönhetően a Minix egyike a legjobban átgondolt felépítésű operációs rendszereknek.) Tehát a Minix tervezői nem akartak ilyen „durva” megoldásokat alkalmazni, ezért a következőt találták ki: bizonyos – az óramegszakítással kapcsolatos – feladatokat maga az operációs rendszer megszakításkezelője intéz el. Ilyen feladat például a pillanatnyi futó folyamathoz rendelt számláló értékének csökkentése. Az órakezelőnek csak abban az esetben küld üzenetet (tehát csak akkor fog meghívódni), amikor valamilyen összetettebb feladat elvégzésére van szükség (lásd később).

Láthatjuk tehát, hogy bizonyos esetekben a feladatokat érdemes úgymond szétosztani az operációs rendszer különböző részei között. Jelen esetben ez a megosztottság több mint 15 százalékos sebességnövekedést jelent, ami az operációs rendszerek körében rendkívülinek számít.

A processzoridő könyvelése és a felügyeleti időzítők

A továbbiakban két nagyon hasznos, szintén az órakezelőre háruló feladatot mutatunk be. Az első az úgynevezett processzoridő-könyvelés, ami azt jelenti, hogy nyilvántartjuk, melyik folyamat mennyi ideig futott (azaz mennyit használta a processzort). Erre kétféle módszer kínálkozik: egy pontos és egy kevésbé pontos. Az első esetben minden folyamathoz rendelünk egy időzítőt, és amikor a folyamat megáll (vagy felfüggesztődik), a tartalmát mentjük, és amikor ismét futni kezd, visszatöltjük.

A másik módszer az, hogy a folyamattáblában egy bejegyzést készítünk, ami a futási időt fogja tárolni. Amikor óramegszakítás érkezik, megkeressük az éppen futó folyamatot a folyamattáblában, majd a futásidőt tartalmazó bejegyzését eggyel növeljük. Ez az elsőnél jóval egyszerűbb módszer, de sajnos akad egy hátránya: amikor a processzor a beérkező megszakításokkal foglalkozik, az is a folyamat „számlájára íródik”. Tehát nem a tiszta futási időt kapjuk vissza, hanem azt is, amikor például a lemezkezelő egy beérkezett megszakítás kezelésével volt elfoglalva. Ez azzal jár, hogy ha sok ilyen megszakítás történik, akkor nem kapunk pontos eredményt.

Némelyik operációs rendszer rendelkezik egy ***Profiling*** nevű szolgáltatással (amit futásidő-elemzésként fordíthatnánk). Ez azt jelenti, hogy nemcsak az egész programnak, hanem az egyes részeinek a futásidejét is mérhetjük.

Az órakezelő másik hasznos szolgáltatása, hogy minden folyamat kérhet tőle figyelmeztetést (más néven riasztást vagy időzítést), ha egy meghatározott idő eltelik. Ez a gyakorlatban úgy működik, hogy a folyamat beállít az órakezelőnek egy időpontot, és amikor az elmúlt, egy figyelmeztetést kap – ez lehet egy üzenet, egy megszakítás vagy akár egy jel (ez rendszerenként különböző).

Lássunk néhány példát

Mire lehet ezt használni? Vegyük a legegyszerűbb példát, mondjuk egy vizsgáztató alkalmazást. Ez a felhasználónak (jelen esetben a vizsgázónak) feltesz egy kérdést, amelyre az adott időn belül válaszolni kell, ezért a program kér egy figyelmeztetést a kérdés megválaszolásához rendelkezésre álló időre. Ha a felhasználó közben választ ad, akkor törli a figyelmeztetést, majd megnézi, hogy helyes-e a felelete. Ha a figyelmeztetésig nem érkezik válasz, elárulja a megoldást, majd a következő kérdésre ugrik.

Egy másik jellemző példa riasztás használatára a hálózati alkalmazások működése. Vegyük azt az esetet, amikor egy másik gépnek elküldünk egy csomagot, és megerősítést (nyugtázást) várunk arról, hogy az általunk küldött csomag megérkezett-e. Ilyenkor beállíthatunk egy figyelmeztetést, és ha a nyugta a riasztás idejéig nem érkezett meg (mert például az általunk küldött csomag elveszett, vagy maga a nyugta tűnt el valahol idefelé jövet), akkor a csomagot újra kell küldenünk.

Érdekes megnézni, hogy az órakezelő miképpen valósítja meg a figyelmeztetéseket. Mint már említettük, a számítógépben lévő programozható órameghajtó általában több, egymástól független órát tartalmaz, amelyeket külön-külön használhatunk. Ha tehát van elég óránk, nincs más dolgunk, mint minden egyes figyelmeztetéshez beállítani egyet, hogy amikor a megadott idő letelik, megszakítást hozzon létre.

Sajnos ennyi óra csak a legritkább esetekben áll rendelkezésünkre, ezért az órakezelőnek több órát kell utánoznia. Erre sokféle módszer kínálkozik, de általában a következőt használják: a kért riasztásokat egy táblázatba rendezik. A táblázatból kikeresik azt az időpontot, amelyik a leghamarabb fog bekövetkezni, és ezt egy változóban tárolják. (Ezt a műveletet mindig meg kell ismételni, valahányszor csak új riasztási kérelem érkezik be). Amikor a pontos idő módosítására kerül a sor (tehát nem minden óramegszakításkor), az órakezelő megnézi, hogy ez az időpont elérkezett-e már. Ha nem, akkor nem történik semmi, ha igen, akkor elküldi a figyelmeztetést a megfelelő folyamatnak, majd kikeresi a következő riasztás időpontját, és kezdődik minden elölről.

A felügyeleti időzítés

A figyelmeztetés különleges formája a felügyeleti időzítés (watchdog timing). A különbség annyi, hogyha a megadott idő letelik, akkor nem egy jel (signal) vagy egy üzenet (esetleg megszakítás) történik, hanem egy előre meghatározott eljárás hívódik meg. A felügyeleti időzítőket az eszközmeghajtók használják.

Két kérdés merülhet fel. Az első (erre könnyebb válaszolni): mikor lehet szüksége egy eszközkezelőnek időzítésre? Ehhez sorozatunk előző részét érdemes megnézni, amelyben a hajlékonylemez-meghajtókról írtunk: megemlítettük, hogy a hajlékonylemez-kezelő bizonyos biztonsági intézkedéseket kap (például a kért blokk a megadott idő alatt a lemez hibája miatt nem érkezik meg, és a meghajtó többszöri próbálkozásra sem tudja beolvasni). A felügyeleti időzítés például ilyen biztonsági intézkedés. A lemezkezelőhöz tartozik egy eljárás, ennek címe adódik át az órakezelőnek. Ha a beállított idő lejár, az órakezelő önműködően meghívja ezt az eljárást, ami már „tudja”, hogy a kért művelet nem járt sikerrel, és megszakítja az olvasási folyamatot.

De felügyeleti időzítést használunk akkor is, amikor például a hajlékonylemez motorját kapcsoljuk be. Mint már említettük, amikor nincs lemezművelet, a motor ki van kapcsolva, hogy ne kopjon a lemez. Amikor kérés érkezik, először be kell kapcsolnunk, ám ez időbe telik, csaknem fél másodpercbe. Addig a lemezkezelőnek unatkoznia kell, tehát kér egy időzítést fél másodpercre, és blokkol. Láthatjuk tehát, hogy a felügyeleti időzítés a B-K eszközök programozásánál szinte nélkülözhetetlen.

A másik kérdés az, hogy az eszközkezelők miért felügyeleti időzítést alkalmaznak, és nem „hagyományosat”. Erre az a válasz, hogy ezek a rendszermagon belül futnak. A Minix például jelek formájában küldi a folyamatoknak a figyelmeztetést. Rendszermagon belül azonban nincsenek jelek. A másik indok az, hogyha mégis üzenetet vagy valami hasonlót küldünk, nem biztos, hogy azt a eszközkezelő megkapja vagy kezelni tudja. Elképzelhető, hogy az eszközben lefagyás történt, és addig meg se moccan, amíg az általa kezelt eszköztől választ nem kap. Ezért sokkal biztonságosabb, ha közvetlen eljáráshívást hajtunk végre, így a meghívott eljárásnak lehetősége lesz életre kelteni a kezelőt, és kiírni a hibaüzenetet. (Ezzel egy kicsit megcáfoltuk az előzőekben tett állításunkat, miszerint a Minix folyamatokra épülő rendszer, így tartózkodni kell a közvetlen eljáráshívásoktól. Sajnos vannak helyzetek, amikor ki kell egyeznünk, és bizonyos engedményeket kell tennünk. Ez is egy ilyen helyzet. Ha ebben az esetben sem engednénk meg a közvetlen eljáráshívást, más módszerre lenne szükségünk, például egy megszakításra, amellyel ezt helyettesíteni tudnánk. De a rendszermagon belüli megszakítások elég veszélyesek lehetnek. Kereshetünk valami kevésbé durva megoldást is, viszont annak megvalósítása annyi munkával járna, hogy nem érné meg.)

Ezzel végére is értünk a beviteli–kiviteli eszközkezelés leírásának. Ezek után elhagyjuk az eszközszintet, és a memóriával kezdünk el foglalkozni, illetve az operációs rendszer úgynevezett memóriakezelőjével. Ebbe a témakörbe rengeteg minden beletartozik, például a virtuális memóriakezelés, az osztott könyvtárak (libraries) működése; de azt is követni fogjuk, hogy mi történik akkor, ha egy alkalmazást Linux alatt elindítunk. Ezzel párhuzamosan a fájlrendszerrel is megismerkedünk, amely a Unixok esetében igencsak központi szerepet játszik. Ugyanis a fájlrendszer nemcsak az adataink tárolásáért felelős, hanem azért is, hogy a folyamatok a szükséges erőforrásokat el tudják érni, illetve ellenőrizze, hogy minden felhasználó csak azt tegye, ami számára engedélyezett (tehát a jogosultságok ellenőrzése is ide tartozik).

Ezek a területek az operációs rendszernek azok a részei, amelyet a felhasználó már közvetlenül érzékel. Míg az eszközkezelés a legtöbb rendszerben többé-kevésbé ugyanazon az elven működik, addig az előbb említett szolgáltatások megvalósítása rendszerfüggő lehet. Ezért most igyekszünk már nemcsak általánosságokban beszélni az operációs rendszerekről, hanem gyakorlati példákat is megemlítünk. Továbbra is elsősorban a Unix alapú rendszereknél (elsősorban a Linuxnál) maradunk, de összehasonlításképpen kitekintünk más „létsíkokra” is, például a Windows NT/2000/XP-k felé is.

Témánk tehát akad bőven, így a következő részben neki is kezdünk. Az első, amivel foglalkozni fogunk, a memóriagazdálkodás lesz.

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.

OpenACS-sablonok

Reuven bemutatja, hogyan gyűjtenek és adnak vissza adatot, valamint végeznek önműködő hibaellenőrzést az OpenACS sablonjai.

Ebónapban az OpenACS sablonozórendszerét vesszük közelebbről szemügyre (lásd még a ➲ http://openacs.org/doc/openacs-4/templates.html címet), ami sok tekintetben hasonlít a Zope lapsablonjaihoz (Zope Page Templates – ZPT). Az OpenACS sablonjai kicsit kifinomultabbak, hiszen képesek adatokat gyűjteni és visszaadni, illetve lehetővé teszik olyan önműködő hibakezelési feladatok egyszerű megoldását, amelyet egyébként igen nehéz lenne megvalósítani, vagy egyszerűen kihagynánk.

Az OpenACS-sablonok

Az ArsDigita programozói úgy határoztak, itt az ideje a paradigmaváltásnak. Többé nem hívunk meg lapokat a szokásos ***.html*** és ***.adp*** kiterjesztésekkel; helyette ezentúl mindent kiterjesztések nélkül hívunk meg. Ez azért volt megvalósítható, mert az AOLserver képes kikeresni a megfelelő lapot. A */foo* URL esetében például az AOLserver először a */foo.tcl*, majd a */foo.adp* végül a */foo.html* lapokat próbálja megnyitni.
Az OpenACS sablonozórendszer ezt a képességet használja ki, hogy a munkát megoszthassa a két különböző fájl között.
A HTTP-kérelmek létrehozta kimenet általában két különböző fájlon megy át. Elsőként a ***.tcl*** fájl hajtódik végre, elvégezve az adatbázis-lekérdezéseket és beállítva a változókat. Az utolsó sora általában az `ad_return_template` szokott lenni, egy Tcl-függvény, ami a hozzá tartozó ***.adp***-lapot hívja meg. Az ADP a változókat adatforrásként kapja meg.
Minthogy a ***.tcl***- és az ***.adp***-lapokat várhatóan különböző emberek fejlesztik, természetesnek tűnik, hogy nagyon szétválnak, vagy együttműködési nehézségeik lesznek. Az ArsDigita mérnökei ezt a gondot a „page contract” (lapszabvány) létrehozásával próbálták elkerülni. A lapszabvány a ***.tcl*** fájl által várt HTTP-értékek listája, illetve olyan Tcl-változókészlet (ezeket adatforrásnak – data source – nevezzük), amelyeket az ***.adp***-lap megjelenítésekor használunk fel.

Adatforrások

Az adatforrások egyszerű Tcl-változók, csak másképp nevezzük őket. Az ***.adp***-lapon belül az adatforrásokra `@nØv@` formátumban hivatkozhatunk, azaz a változónév köré @ jeleket helyezünk. Amennyiben az adatforrás nincsen meghatározva, a sablon Tcl-veremlenyomat (stack trace) kiírásával leáll, sérelmezve az ismeretlen változót.
Az adatforrások a fájlon belül bárhol megjelenhetnek. Az értékek még azelőtt eljutnak a HTML-lap megfelelő helyére, mielőtt az a felhasználó böngészőjére kerülne, így az adatforrások nemcsak szöveget, hanem képneveket vagy stíluslapértékeket is megadhatnak. Például az alábbi egyszerű OpenACS-sablon a fejlécsorban a felhasználó vezetéknevét jeleníti meg:

```
<master>
     <h2>@first_name@</h2>

     <p>That's you, isn't it?</p>
</master>
```

A `master` tag jelzi, hogy a kérdéses lap nem teljes értékű HTML-kimenet, hanem a helyi mesterlapba kell beszúrni (azaz a ***master.tcl/master.adp*** lappár által meghatározott lapba). A helyi mesterlap pedig a webhely alapértelmezett mesterlapjába (ez általában a ***default-master.tcl*** és ***default-master.adp*** nevű lappár, de állítható jellemzőről van szó) kerül majd be. Így aztán az eredménylapon a következők jelennek meg:

```
az alapØrtelmezett mester teteje
     a helyi mester teteje
          a sajÆt lapunk
     a helyi mester alja
az alapØrtelmezett mester alja
```

Ezzel a módszerrel könnyen készíthetünk egységes kinézettel rendelkező lapokat azonos fej- és lábléccel, amely például menüsort és kapcsolatadatokat tartalmazhat. A mester és az alapértelmezett mesterlapok jelölése sok tekintetben hasonló a Perl alapú `HTML::Mason` sablonozó rendszer autohandlerjeihez.
Ez mind nagyon szép és jó, de hol van a mi `@first_name@` változónk meghatározva? Nos, ezt a ***.tcl***-társlapban tehetjük meg. Csakhogy nem elég egyszerűen a ***.tcl*** fájlban a `first_name` változót beállítanunk. Meg is kell jelölnünk exportálandó adatforrásként azáltal, hogy közvetlenül megnevezzük.
A ***.tcl***-lapok kimenetüket és bemenetüket az `ad_page_contract` függvény értékeiben adhatják meg. Ez többnyire a lap tetején történik meg. Az `ad_page_contract` hatékony módszer olyan lapok létrehozására, amelyek bizonyos bemenetet várnak, és cserébe várhatóan adott kimenetet készítenek. Az `ad_page_contract` meghívása a hihetetlen egyszerűtől a hihetetlen összetettig változhat a ***.adp***-lapsablon igényeinek megfelelően. Például egy a felhasználó nevét valami próbaértékre beállító, majd azt exportáló ***.tcl***-lap valahogy így nézne ki:

```
ad_page_contract {
    A megjegyzØsek Øs a CVS-adatok ide ker lnek.
} {
} -properties {
    first_name:onevalue
}

set first_name "Pr ba keresztnØv"

ad_return_template
```

Az `ad_page_contract` hívás mutatja meg a sablonozórendszernek, hogy a `first_name` változót exportálni fogjuk. Ezután beállítjuk a `vÆltoz` értékét, majd ezt az értéket az `ad_return_template` hívással átadjuk a sablonnak. (Sajná-

latos történelmi okokból kifolyólag valamilyen jellemzőbb név helyett az adatforrások a `-properties` változón keresztül kerülnek visszaadásra.)
Több változót is átadhatunk a sablonnak, ha további sorokat írunk az `ad_page_contract`-ba:

```
ad_page_contract {
    A megjegyzØsek Øs a CVS-adatok ide ker lnek.
} {
} -properties {
    first_name:onevalue
    last_name:onevalue
}

set first_name "FirstName"
set last_name "LastName"

ad_return_template
```

Listák és sorkészletek (multirow)

Mint láthattuk, az `ad_page_contract` segítségével egyszerűen adhatunk át szöveges adatot a sablonnak. Sok esetben azonban, főként ha adatbázisból kérünk le adatot, értékek listáját szeretnénk átadni. Az OpenACS-sablonok ezt is gond nélkül képesek kezelni. Példaképpen a következő ***.tcl***-lap a rendszer felhasználóinak listáját kéri le és helyezi a „list” adatforrásba:

```
ad_page_contract {
    A megjegyzØsek Øs a CVS-adatok ide
ker lnek.
 } {
 } -properties {
    users:onelist
}

 set users [db_list get_all_users {
    SELECT PE.first_names || ' ' ||
           PE.last_name as users
      FROM Parties PA, Persons PE
     WHERE PA.party_id = PE.person_id
  ORDER BY PE.last_name, PE.first_Names }]

 ad_return_template
```

Láthatjuk, hogy SQL-lekérdezésünk egyetlen ***users*** nevű oszlopot ad vissza. Az OpenACS adatbázis ***felülete*** ezt a ***users*** Tcl-listává alakítja, amit aztán ***users*** adatforrásként exportálhatunk. Mivel a `:onelist` leíróval exportáltuk, az ***.adp***-lap az összes elemén végiglépkedhet:

```
<master>
      <list name="users">
          <li> @users:item@
      </list>
</master>
```

A `<list>` ismétlőszerkezeten belül a benne foglalt tartalom a lista minden egyes elmére végrehajtódik. Az éppen időszerű elemet `@users:item@` formában érhetjük el; a pillanatnyi ciklussorszámot a `@users:rownum@` tartalmazza, végül a jelenlegi ismétlés `@users` formában érhető el.
Amennyiben több adatbázissoron keresztül szeretnénk ismételgetni, a ***.tcl***-lap sorkészleteket (multirow) is képes exportálni.

A sorkészlet az összes visszaadott sort tartalmazza, az oszlopnevekkel együtt. A dolgok Tcl-oldala a következőképpen néz ki:

```
ad_page_contract {
} {
} -properties {
    users:multirow
}

db_multirow users get_info "
    SELECT PE.first_names || ' ' ||
           PE.last_name as name,
            PA.email FROM Parties PA, Persons PE
     WHERE PA.party_id = PE.person_id
  ORDER BY PE.last_name, PE.first_names"

ad_return_template
```

A `db_multirow` függvény három értéket vár: a benépesítendő tömb nevét (amelyet majd adatforrásként exportálunk), a lekérdezés nevét (amit az adatbázis-független ***.xql*** fájllal kiegészítve használhatunk) és a tartalék lekérdezést, amelyet akkor használunk, ha az ***.xql*** fájl nem található. Az ***.adp*** sablonban a következőket írjuk:

```
<master>
  <ul>
  <multiple name="users">
     <li>
     <a href="mailto:@users.email@"
     @users.name@</a>
  </multiple>
  </ul>
</master>
```

Két trükkös dologra hívnám fel a figyelmet. Először is az ismétlő címke neve multiple, annak ellenére, hogy az adatforrást multirow-ként (sorkészlet) exportáltuk. Ha rossz nevet használunk rossz helyen, az bizony nehezen felderíthető hibákhoz vezethet. Még szebb, hogy a `<multiple>` tagokon belüli elemleíró jel a pont (`@users.email@`), holott a `<list>` tagokban kettőspontot használtunk (`@users:item@`). Jómagam folyamatosan elkövetem ezeket a hibákat, és mindig a korábbi működő lapokat nézegetem, hogy lássam, a helyes írásmódot használom-e a megfelelő lapon.

Bemenetek

Eddig csak azzal foglalkoztunk, hogyan adhatunk át az `ad_page_contract` segítségével adatokat a ***.tcl***-lapból az ***.adp***-lapnak. A ***.tcl***-lapok GET- vagy POST-kérelmeken keresztül azonban bemeneteket is tudnak fogadni. Az `ad_page_contract` lehetővé teszi, hogy megadjuk, milyen bemeneteket várhatunk, szükség szerint alapértelmezett értékeket rendeljünk hozzájuk, sőt ellenőrizzük, hogy a bemenet a megfelelő formátumban van-e:

```
ad_page_contract {
} {
    foo
} -properties {
    foo2:onevalue
}

set foo2 "$foo$foo"
```

```
ad_return_template
```

A fenti példában a *.tcl*-lap `foo` nevű értéke várja (akár GET, akár POST üzeneten keresztül a bemenetet). Az értéket aztán a `foo2` nevű új adatforrás létrehozására használja fel, amelybe a `foo` kétszerezett értéke kerül. A `foo`-hoz alapértelmezett értéket is rendelhetünk, ha a Tcl-lista első elemévé tesszük, majd az alapértelmezett értéket a második elembe helyezzük:

```
ad_page_contract {
} {
    {foo "blah"}
} -properties {
    foo2:onevalue
}

set foo2 "$foo$foo"

ad_return_template
```

Ha ezután a lapot a `foo=abc` értékkel hívjuk meg, az „abcabc" karaktereket kapjuk vissza, ha érték nélkül hívjuk meg, akkor „blahblah" lesz az eredmény.
Minden értékhez egy vagy több jellemzőt is rendelhetünk, így korlátozva a bekért adattípusokat. Például levághatjuk a bevezető és befejező szóköz karaktereket a bemenő értékről, vagy biztosíthatjuk, hogy nullánál nagyobb egész számot kapjunk:

```
ad_page_contract {
} {
    sometext:trim
    anumber:naturalnum
}
```

Egy értékben több jellemzőt is használhatunk, ha vesszővel választjuk el őket egymástól:

```
ad_page_contract {
} {
    sometext:trim,nohtml
    {anumber:naturalnum 50}
}
```

A fenti lapszabvány azt mondja: az `anumber`-nek természetes számnak kell lennie, ha pedig nincsen megadva, az alapértelmezett értéke 50. A `sometext` elejéről és végéről el kell távolítani a szóköz karaktereket, és nem tartalmazhat HTML-tagokat. Létezik egy `html` és egy `allhtml` jellemző is, amelyek biztonságos HTML-címkekezelést, illetve az összes HTML-címkét elérhetővé teszik. A lapszabvány még ennél is többet tud. Létrehozhatunk például egy dátumválasztó elemet (widget) az `ad_dateentrywidget` függvénnyel. Így létrehozható a következő *.tcl*-lap:

```
ad_page_contract {
} {
} -properties {
    datewidget:onevalue
}

set datewidget [ad_dateentrywidget datewidget]

ad_return_template
```

A hozzá tartozó *.adp*-lap, amely a dátumelemet jeleníti majd meg, így néz ki:

```
<master>
    <form method="POST" action="date-2">
    @datewidget@
    <input type="submit" value="Send the date">
    </form>
</master>
```

Más szavakkal a HTML-űrlap a dátumelem tartalmát a `date-2`-be küldi, amely tulajdonképpen az eredményt *.adp*-lappal megjelenítő *.tcl*-kód. A *date-2.tcl* az `ad_page_contract`-ban megadhatja, hogy a bejövő `datewidget` érték egyszerű tömböt tartalmaz. Megadhatjuk azonban a `datewidget`-et date típusúnak is, így önműködően négyelemű tömböt kapunk:

```
ad_page_contract {
} {
  datewidget:array,date
} -properties {
  date_month:onevalue
  date_day:onevalue
  date_year:onevalue
  date_full:onevalue
}
  set date_month $datewidget(month)
  set date_day $datewidget(day)
  set date_year $datewidget(year)
  set date_full $datewidget(date)

  ad_return_template
```

Az időadatokat *.adp*-lapunkon (`date-2`) különböző formátumokban jeleníthetjük meg:

```
<master>
    <p>Month: @date_month@</p>
    <p>Day: @date_day@</p>
    <p>Year: @date_year@</p>
    <p>Full text: @date_full@</p>
</master>
```

Megjegyezzük, hogy a dátumelem (date widget) az SQL-lekérdezésekben is tökéletesen elfogadható. Ez jól jöhet, ha dátumokat kell bevinni, vagy ha összehasonlító lekérdezésekben használjuk őket.

Összegzés

A velem dolgozó grafikusok nagyra értékelik a *.tcl*- és *.adp*-lapok szétválasztását. Míg az OpenACS tanulási görbéje elég meredek lehet, a sablonok működésének megértése egyszerre örömteli és érdekes módja a rendszer megismerésének.

Linux Journal 2003. január, 105. szám

Reuven M. Lerner (reuven@lerner.co.il)
Egy kisebb webes és internetes módszerekkel foglalkozó tanácsadó cég tulajdonosa és vezetője. Az ATF honlapon érhető el (➲ http:/www.lerner.co.il/atf/).

Ha majd felhívlak... a képtelefonomon

Megvalósult végre a jövő egyik ígérete? A GnomeMeeting ennek kezdetét jelzi.

Emlékszel, François, a 2001 – Űrodüsszeia című filmre? Így, 2003 küszöbén nem tudom megállni, hogy ne gondolkozzam el egy kissé azon a kilátáson, amit a jövő e filmbéli látomása tartogatott számunkra, és ez egy kicsit elszomorít, mon ami. Bár gondoljuk csak végig! Van már űrállomásunk, nem teljesen olyan, mint az a méltóságteljes kerék, amit a filmben megcsodálhattunk. Természetesen nincs holdbázisunk, sem földkörüli szállodánk, de legalább itt van nekünk ez. Quoi? Hiszen ez egy videotelefon François, és ma éppen ezt szeretném vendégeinknek bemutatni. Hogy mondod? Á, már itt is vannak! Köszöntelek titeket ismét, mes amis, Chez Marcelnél, ahol a kitűnő Linux-konyha mellett az ízletes borokat is megtalálhatjátok.

Ebben az étteremben korábban már megismerhettetek olyan recepteket, amelyek webkamerátok használatában nyújtanak némi segítséget, a képrögzítéstől kezdve az otthoni biztonsági rendszerig. Ma ismét használni fogjuk a kamerát, méghozzá olyasvalaminek a megvalósítására, amit már annyi éve ígérgetnek nekünk. Amikor még kisgyerek voltam, minden televíziós tudományos-fantasztikus műsor mindennapos használati tárgyként mutatta be a képtelefont. Az évek egyre múltak, és bár a tévében látni lehetett ilyen eszközöket, a saját képtelefon-készülék olyan távoli maradt, mint a televízióstúdió maga. Az étterem megnyitása óta örömmel tapasztalhattam, hogy számos gyerekkori álmom valóra vált a Linuxszal történő főzés során. Ilyen a képtelefon is, illetve annak egy használható megvalósítása, ***Damien Sandras*** GnomeMeeting nevű programja. A GnomeMeeting együtt tud működni a Microsoft Netmeetinggel is. A GnomeMeeting KDE alatt is kitűnően működik, nagyszerűen illeszkedik a pulthoz. Számos terjesztés tartalmazza a GnomeMeetinget és a szükséges programkönyvtárakat (`pwlib` és `openh323`), bár nem biztos, hogy telepíti is őket. A ➲ http://www.gnomemeeting.org honlap a fontosabb terjesztésekhez előre fordított csomagokat (RPM-eket és DEB-eket) kínál, ugyanitt a forrást is megtalálhatjuk (44. CD Magazin/gnomeeting). Érdemes e helyütt kezdeni a kutakodást. Az OpenH323 csomag PwLib-függő, ezért ezt isfeltétlenül telepíteni kell. Arra azért felhívom a figyelmeteket, hogy bár a GnomeMeeting forrásból is lefordítható, az OpenH323 könyvtárak telepítése meglehetősen bonyolult lehet (nem is beszélve időigényességéről). Ha ehhez bináris, előre elkészített állományokat–csomagokat tudtok használni, mindenképpen javaslom a használatukat. Amikor első alkalommal futtatjuk a programot, rögtön megismerkedhetünk a ***Beállításvarázslóval*** (First Time Configuration Druid). Ez a beállítási folyamat a GnomeMeeting felhasználói könyvébe való bejegyzést is magában foglalja – ezt úgy is elképzelhetjük, mint egy nagy, hálózaton keresztül elérhető telefonkönyvet. A varázsló regisztrációs ablakának alján található jelölőnégyzet segítségével dönthetünk úgy is, hogy ne szerepeljünk a listán. Ha elégedettek vagyunk a beírt adatokkal, a ***Forward*** (Tovább) gombbal léphetünk a következő beállításra, ami a használt kapcsolat típusának megadását kéri.

A befejező ablak ***Apply*** (Alkalmaz) feliratú gombjára kattintva elindul a program. Számos beállítást megváltoztathatunk, ha a menüsor ***Edit*** (Szerkesztés) pontjának ***Preferences*** (Beállítások) tagját választjuk. Később bármikor visszatérhetünk a varázslóhoz. Az általam megváltoztatott első beállítás a ***Video Devices*** (képkezelő eszközök) rész alatt található. Azt szerettem volna, ha a GnomeMeeting indulásakor az előnézeti kép működik, és nagyméretűre van állítva. Ha a helyi hálózaton keresztül szeretnék hívást kezdeményezni egy másik számítógép felé, beírom a `callto://192.168.22.2` karaktersort a GnomeMeeting címsávjába, közvetlenül a menüsor alatt. Egy kis előugró ablak jelenik meg a másik számítógép kijelzőjén, figyelmeztetve a felhasználót a bejövő kapcsolatra. Ha ezt a felhasználó elfogadja, a két gép máris kész az összeköttetésre.

A *2. kép* bal oldalán megfigyelhető az a gombsor, amelynek a segítségével be- és kikapcsolhatjuk a képet és a hangot,

A GnomeMeeting beállítása szinte gyerekjáték

Készen állunk arra, hogy a GnomeMeetinggel hívjuk a világot

szöveges üzenetváltáshoz pedig csevegőablak megnyitására is lehetőségünk nyílik. Ha rendelkezünk mikrofonnal, a GnomeMeetinggel ennek is jó hasznát vehetjük. A program számos beállításban működni tud, így csak kép, csak hang vagy csak szöveges üzemmód vagy ennek a háromnak tetszőleges kombinációja is rendelkezésünkre áll – bár igen zavaró lehet a tudat, hogy valaki lát minket, de mi nem látjuk őt. A GnomeMeeting futtatásakor feltétlenül kapcsoljuk be a vezérlőpultot (***Control Panel***), ami a program egy fülekkel ellátott ablakában felelnek meg, helyet adva a hang és kép beállítására szolgáló eszközöknek és az előzményeket mutató ablaknak. Ez utóbbiban láthatók a hívások állapotai, a hálózaton keresztül elérhető könyvtáraink és az egyéb adatok. Az ablak bármikor anélkül ki-bekapcsolható, hogy ez az adatátvitelt befolyásolná. Az űrállomásról kiinduló hívás igen nagy távolságokat szelt át: egy földön

Valaki ebédre hív magához

élő kislánynak szólt. Bár az irodai csevegések is igen szórakoztatóak, felmerül a kérdés, hogy mi a helyzet a külvilággal. Kit hívhatnánk fel? A GnomeMeeting hivatalos telefonkönyve a ➲ http://ils.seconix.com címen érhető el. Ahhoz, hogy a könyvben böngészhessünk és kikeressük a GnomeMeeting többi felhasználóját, magunkat is be kell jegyeztetnünk. Indítsuk el a GnomeMeetinget, kattintsunk a telefonkönyv ikonjára a bal oldalon, és már kereshetjük is a beszélgetőpartnerünket – ilyen egyszerű… legalábbis majdnem.

Nos, ha az otthoni vagy vállalati hálózaton futtatjátok a GnomeMeetinget, nem lehet semmi gond. Ugyanez igaz akkor is, ha egy önálló gépen futtatjuk a programot, ami a Világhálóhoz kapcsolódik – jó eséllyel működni fog. Kelepcébe akkor esnünk, mes amis, ha egy álcázó (masquerading) illetve NAT-ot használó (Network Address Translation – hálózati címfordítás) tűzfal mögül próbáljuk a GnomeMeetinget működésre bírni. Az alábbi kapuknak kell ehhez engedélyezettnek lenniük:

```
TCP: 1720, 30000-30010
UDP: 5000-5003
```

A játéknak ezen a pontján a tűzfalak működésének meglehetősen bonyolult témájába botlunk. Én hamar eljutottam odáig, hogy csak a NAT egy másik lehetséges helyettesítője, a RSIP for Linux segítségével juthattam tovább, ami ezt a nehézséget egyszerűen közelíti meg. A RSIP egy viszonylag új protokoll, ami a NAT-hoz hasonlóan azt teszi lehetővé, hogy egy kapcsolatot több ügyfél között megosszunk. Böngészhetjük a Hálót, leveleket küldhetünk, bármit megtehetünk, ami az ügyfelek kiszolgálásának a körébe tartozik. Egy tűzfal mögött lévő kiszolgálón egy kapu átirányítására is könnyen használhatjuk anélkül, hogy a csomagok bármilyen változást szenvednének. A NAT vagy az álcázás használatakor ez csak betöltődő modulokkal valósítható meg. Ezek a modulok legfeljebb kipróbálásra valók, amikor a H323 vagy jó barátunk a GnomeMeeting használatára kerül sor. A RSIP megoldja ezt a gondot, ami természetesen nem azt jelenti, hogy (IP-Tables-beállításainkat) el kellene dobnunk – a RSIP is jó hasznát veheti ezeknek.

Első lépésként töltsük le a legfrissebb forrást a ➲ http://openresources.info.ucl.ac.be/rsip/index.php címről. A legtöbb terjesztéshez előre fordított bináris állományokat is találunk, de ha nincs számunkra megfelelő, akkor se essünk kétségbe, a fordítás ezúttal igen egyszerű:

```
# tar -xzvf rsipd-
  ➥0.9.3.src.tar.gz
# cd rsipd-0.9.3
# make
# su -c "make install"
```

Ezzel a kiszolgálóoldali telepítéssel el is készültünk, de az ügyféloldali részre is szükségünk van. Ez két változatban érhető el: egy Python nyelven írott programként és egy rendszermagmodul formájában. A menü összeállításának idején a modul inkább kipróbálásra alkalmas, míg a Python-változat üzembiztosnak volt mondható. A modul telepítése a forrás kibontásából és egy `make install` parancs futtatásából áll, a dolog hátulütője viszont, hogy csak a 2.4.18-as és a 2.4.19-es rendszermagokat támogatja. Python-ügyfél esetén nincs szükség programfordításra.

A RSIP GnomeMeetinggel való használatához először nyissuk meg a */etc/rsip/rsipd.conf* fájlt, és módosítsunk néhány fontos értéket. Ezek: a tűzfalunk belső és külső csatolófelületei és a kiszolgáló által kezelt hálózati címtartomány. Én a `192.168.22.0` című belső hálózat első tíz címét foglaltam le a célra:

```
EXT_IF eth1
INT_IF eth0
POOL 192.168.22.1-10
```

Most indítsuk el a kiszolgálót az `install` által létrehozott parancsfájl segítségével:

```
/etc/init.d/rsipd start
```

A Python-ügyfél futtatásához az alábbi parancsot használhatjuk (figyeljünk arra, hogy ez egyetlen hosszú sor; a `-s` kapcsoló által megadott IP-cím a kiszolgáló címe):

```
# python /path_to/rsipclient-
➥0.19.py
➥-d -v -s 192.168.22.10
➥-l 1720,5000,5001,5002,5003,
➥30000,3000,30002,30003,30004,
➥30005,30006,3000,
➥30008,30009,30010
```

Talán meg kellett volna jegyeznem, hogy az ügyfélprogram az ügyfélgépen fut, non? Ezt szintén rendszergazdaként kell futtatnotok. A dolog szépsége, hogy a korábbi tűzfal-beállításokat tovább használhatjátok. A rendszermagmodul használata még egyszerűbb. Mindössze arra van szükség, hogy a kiszolgáló címének megadásával betöltsük:

```
# insmod krsip ip=192.168.22.10
```

Próbaképpen a bal oldali ikon segítségével megnyitottam az ILS-kereső ablakot. Ennek hatására egy XDAP-kiszolgáló böngésző jelent meg, amelyben a ***Refresh*** (frissítés) gombra kattintottam. Nagyjából véletlenszerűen kiválasztottam valakit, és kettőt kattintottam a bejegyzésére. Pár másodperccel később kanadai otthonomból már javában csevegtem párizsi partneremmel. Látjátok, ennyi az egész. Máris készen álltok arra, hogy osztozzatok a képtelefonálás izgalmaiban, és csak két év késésben vagytok az előre jelzett időponthoz képest.

A záróra, mes amis, ismét közeleg. Legalább mostanra elértünk oda, hogy cseveghetünk ultramodern képtelefonunk segítségével (még ha 2001 el is múlt már). Már a jövő sem a régi, nem igaz? Erre 2010-ben még visszatérünk. Mielőtt elköszönnék tőletek, François utoljára teletölti poharaitokat. Az utolsókat kortyolgatva nézzük meg, hogy ki van még fenn ilyenkor Finnországban! Viszontlátásra a következő hónapig.

A votre santé! Bon appétit!

Linux Journal 2003. január, 105. szám

Marcel Gagné
(mggagne@salmar.com)
Mississaguában, Ontario államban él. Ő a szerzője a Kiskapu kiadásában szeptemberben megjelent Linux-rendszerfelügyelet (ISBN 963-9301-40x) című könyvnek (jelenleg is egy könyvön dolgozik).

KAPCSOLÓDÓ CÍMEK

GnomeMeeting ➲ http://www.gnomemeeting.org
OpenH323 ➲ http://www.openh323.org
RISP for Linux ➲ http://openresources.info.ucl.ac.be/rsip/index.php
Marcel borlapja ➲ http://www.marcelgagne.com/wine.html

Gombnyomás a távolból

A hálózati eszközök néha lefagynak, ezért újra kell őket indítanunk. Ez nem egyszerű feladat, ha nem vagyunk a készülék közvetlen közelében.

Először is el kell jutnunk a készülékhez, amihez néha nagyon sok kilométert meg kell tennünk, vagy megkérnünk valakit, aki a közelében lehet, hogy nyomja meg a „RESET” gombot. Mindkettő idő-és költségigényes feladat. Ezt a feladatot hivatott ellátni a WTI cég Telnet Power Swithes TPS2 és TPS2CE készüléke.

Mire is jó egy ilyen készülék?

Képzeljük el azt a helyzetet, hogy egy kiszolgálónk egy másik földrészen – na jó, lehet közelebb is – leáll, és senki emberfiát nem tudunk kiverni az ágyából, hogy szívességből ugyan indítsa már újra, mert a cég nem tud nélküle működni. Pontosan a fentebbi helyzetekre találták ki ezeket a „kapcsolókat”. A távolból, egyszerűen az Interneten keresztül bejelentkezhetünk, de ha esetleg nem lenne internetkapcsolat, használhatjuk a modemes betárcsázós bejelentkezés módját is. Bejelentkezve a „kapcsolóra”, bármelyik csatlakoztatott eszköz áramellátását „szabotálhatjuk”, kisebb-nagyobb lelki zavarokat okozva magának az eszköznek, de mindenképpen újraindíthatjuk. A gyakorlatban ez a művelet az eszköz kikapcsolásának és bekapcsolásának felel meg.

A készülék

A Mission Critical Linux Kft. jóvoltából mi egy TPS2CE készüléket próbálhattunk ki, ami csak annyiban különbözik a TPS2-től, hogy 230 V-ról működik.
A csomag tartalma: a TPS2CE készülék, hálózati kábel, soros kábel és angol nyelvű kézikönyv. Ez a készülék két eszköz csatlakoztatására alkalmas, azonban létezik 8 kapus változata is webfarmokhoz, fürtökhöz.

Beüzemelés

Mielőtt bekapcsoljuk, csatlakoztassuk a modemet, a soros kábelt és a hálózati kábelt mind a „kapcsolóhoz”, mind a telefonvonalhoz, a konzolhoz és a hálózathoz (igény szerint bármelyik elhagyható). Bekapcsolás után a „kapcsoló” elején lévő LED-ek jelzik az éppen időszerű állapotot, jelentésük (balról jobbra):

WTI TPS2CE

Gyártó: WTI
Forgalmazó: Mission Critical Linux Kft., Budapest, Frankel Leo u. 11–13.,
Telefon (06-1) 438-41-00
➲ www.missioncriticallinux.hu
TPS-2 CE: 110 000 Ft + áfa
NPS-230 (8-portos) 166 000 Ft + áfa

- ON: világít, ha áram alatt áll,
- RDY: villog, ha a „kapcsoló” kész a parancsok fogadására,
- RXD: világít, ha a „kapcsoló” parancsokat fogad,
- DCD: világít, ha a modemes kapcsolat működik,
- NET: világít, ha a `telnet` kapcsolat működik.

A készülék elején még két LED és kapcsoló található, amelyek szervesen kapcsolódnak egymáshoz. Ezek a LED-ek jelzik a csatlakoztatott eszközök áramellátását, a kapcsolóval pedig ki és be tudjuk kapcsolni az eszközöket.
A legegyszerűbb kapcsolódás a soros konzol kapunján keresztül történhet: a soros kábelt csatlakoztassuk a „készülékhez”, a másik végét pedig egy számítógéphez vagy bármihez, ami soros terminálként tud üzemelni. Bármilyen terminálprogrammal kapcsolódhatunk.
A frissen beszerzett készülékhez történő kapcsolódás másik módja a modemes kapcsolat. Beállítás előtt az ethernetkapcsolatot nem használhatjuk.
Miután a kapcsolat feléledt, egy szinte mindent leíró ablakot kapunk. Ebből az ablakból kétféle létezik: az egyik a rendszer (System), a másik a felhasználói (User) ablak. Első dolgunk a rendszerjelszó beállítása legyen! Fontos megjegyezni, hogy a rendszer a kis- és nagybetűket megkülönbözteti. A rendszer több felhasználói jelszót tud kezelni, amelyekhez hozzárendelhetjük a működtetett eszközöket, ezzel is védve az illetéktelen ki- és bekapcsolgatást.
Ha a rendszer- (System) jelszóval lépünk be, teljes mértékben uraljuk a „készüléket”: mindenfélét állítgathatunk, felhasználókat tarthatunk karban, beállíthatjuk a hálózati kapcsolatot, az IP-címet, hozzáféréseket is korlátozhatunk. Ha felhasználóként lépünk be, akkor a ránk kiszabott jogokkal tehetünk meg mindenfélét. A legfontosabb három parancs, amit én használtam:

- `/ON n`
- `/OFF n`
- `/S illetve /SP`

Az `/ON n` értelemszerűen bekapcsolja, az `/OFF n` pedig kikapcsolja a csatlakoztatott eszköz áramellátását, ahol az `n` az eszköz csatlakozási száma. A `/S` kilistázza a jelenlegi állapotot, a `/SP` pedig rendszermódban a jelszavakat is megjeleníti.

Összegzés

A legcsúnyább dolog, amit ezzel a kis „kapcsolóval” sikerült véghezvinnem, az a saját gépem kikapcsolása volt saját magam alól. Történt ugyanis, hogy soros kapun keresztül jelentkeztem be, és kiadtam a `/OFF 2,Y` parancsot. Gondot az okozott, hogy nem olvastam el a leírást, csak miután leütöttem az ENTER-t. Ez a parancs ugyanis megerősítés nélkül végrehajtódik, mivel a végén ott van az Y. Az eredmény egy kissé hosszadalmas újraindítás, két ideges felhasználó, és ennek a cikknek a 0.5-ös változata. Így tanultam meg az `Y` jelentését a parancsok végén.
Kényelmesen és gyorsan beállítható eszközről van szó, széles határok között lehet a hálózati biztonságot is testreszabni, ami egy ilyen eszköz esetében alapvető elvárás.

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

Kihagyhatatlan kiegészítők és programok a Zaurushoz

Programok, amelyek nélkül el se indulj otthonról, és néhány ötlet, hogyan hozhatod ki a legtöbbet egy Zaurusból.

Gratulálok! Kezedben a Zaurus, kipróbáltad az összes programot, beírtál néhány parancsot a terminálablakba, hogy megbizonyosodj arról, valóban GNU/Linux alapú gépet tartasz-e a kezedben. Nagyjából tudod, hogy melyik gombot merre keresd, bár a csővezeték karakterének a helye még rejtély (tipp: SHIFT+SZÓKÖZ). Merre tovább? Írásomban néhány figyelemre méltó bővítést és alkalmazást ajánlok a figyelmedbe.

Memóriabővítés

A géppel kapott 32 vagy 64 MB memóriánál bizonyára többre lesz szükséged, ugyanis a rendszer ennek túlnyomó részét akkor is elfoglalja, ha még semmit sem telepítettél fel. Egy normál, 32 MB memóriával szerelt 5000d típus 2.37-es ROM-képpel mindössze 600 KB szabad memóriát biztosít, ha semmi sincs telepítve a gépre, és nem is fut rajta semmi. Ha több memóriát akarsz, két lehetőséged van: használhatod az SD/MMC vagy a CompactFlash (CF) foglalatot. A CF-foglalat majd a hálózati kártyához kell, így marad az SD/MMC – természetesen ez sem ilyen egyszerű.

A legnépszerűbb és legolcsóbb SanDisk kártyák gyakran meghibásodnak. A Sharp új illesztőprogramot volt kénytelen kiadni az ötven dollárnál is olcsóbb, 128 MB-os SanDisk kártyák használata kapcsán felmerült bajok miatt, ezek ugyanis felosztás után sokszor használhatatlanokká váltak, vagy épp csak egyetlen használatot bírtak ki. A gond az, hogy a másolásvédelmi rendszer miatt az SD-illesztőprogramok soha nem lehetnek teljesen nyitottak. Vegyél inkább MMC kártyát, ahhoz léteznek szabad illesztőprogramok.

Hordtok és kiegészítők

A Zaurus nagy sláger Japánban, és arrafelé rengeteg kiegészítőt kapni hozzá. Légy tehát nagylelkű önmagaddal szemben, és lepd meg magadat legalább egy Extreme Limit bőrtokkal. A Portfolio nevű modell tökéletes védelmet nyújt a tenyérgépnek. Ugyancsak hasznos darab a Sumdex villámzáras GLP-824 jelzésű iPAQ tokja. Javaslom, hogy vásárolj egy második akkumulátort és egy töltőt is. Ha Zaurusodat dokkolás helyett a tokjában tartod, szükséged lesz egy USB töltő-összehangoló kábelre is. A soros kábelt viszont nem javaslom, pocsékul tervezték, és a billentyűzet kinyitását is akadályozza.

Az SL-5000D és az 5500 frissítése

Az 5500 számos részletben különbözik a múlt évi fejlesztői modelltől, az SL-5000D-től. Az 5500-as modellen Hancom Office is található, jobb fém és műanyag tollakat és váltóáramú adaptert adnak hozzá. Ugyan a Zaurusnak nincs beépített mikrofonja, az utólagos beépítés azonban megoldható (lásd a ***Kapcsolódó címek*** részt).

ROM: Sharp, Opie, Paul vagy Crow?

Most, hogy maga az eszköz készen áll, lássuk a programokat. Számos különféle ROM-lenyomat, röviden: ROM érhető el (lásd a ***Kapcsolódó címek***-et). ROM-lenyomatot a `modzaurus` segítségével létre is tudsz hozni. Ha 2.37-es vagy régebbi Sharp ROM-od van, egy ismertté vált biztonsági hiányosság miatt ajánlott frissíteni.

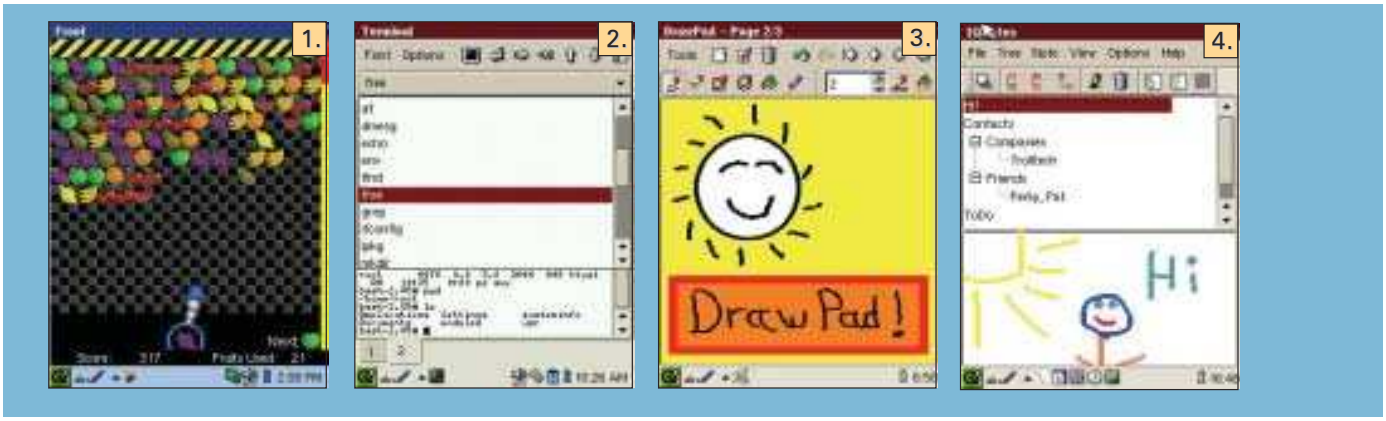
No de melyik ROM-lenyomat a legjobb? A Sharp által készített kerül a gépre alapállapotban, és széles körű támogatást élvez. Az Opie letisztult, szabad ROM-lenyomat, feljavított alapértelmezett alkalmazásokkal. Paul a gyári Sharp ROM-lenyomat, de képes MMC-kártyát tárhelyként használni, így a teljes memória RAM-ként marad használható. A Crow ROM-lenyomat olyan, mint a Paul, Opie-alkalmazásokkal kiegészítve.

Az Opie átgondoltabb tervezésű, fájlrendszere jobban hasonlít a szabványos GNU/Linux fájlrendszerre, és az alkalmazásokból újabb változatokat tartalmaz, mint a Sharp példányai. Hiányzik viszont belőle a Java virtuális gép és az Opera böngésző, és továbbfejlesztett fájlrendszere miatt nem futnak rajta egyes külső programok. Ha van MMC-kártyád, érdemes elgondolkodnod a Paul- vagy Crow-féle változat használatán. Egyszerűen csak formázd meg ext2 fájlrendszerrel az MMC-kártyát, a gép a ***/home*** alatt fogja befűzni. Az Opie is képes MMC-kártya használatára, de vannak bizonyos együttműködési gondjai.

A legjobb ROM-lenyomat tartalmazná a kipofozott, Opie-féle alkalmazásokat, lehetővé tenné a zárt programok telepítését (legalább az Opera, a Jeode, a Hancom Office és a theKompany.com alkalmazások használatát), és természetesen nagyobb baj nélkül működne. Sajnos ilyen lenyomat egyszerűen nem létezik – egyelőre.

A ROM-frissítést óvatosan kell végezni. Először is le kell töltened a kívánt változatot. Ne feledd, nem mindig a legújabb változat a legjobb! Ezután egy USB CompactFlash-olvasóval vagy egy CF-PCMCIA átalakítóval rá kell másolnod egy CF-kártyára. Ügyelj arra, hogy a CF-kártya FAT16 fájlrendszerrel legyen formázva, majd a ROM-lenyomatfájlt nevezd át ***Romimage***-re, és helyezd el a CF-kártya gyökérkönyvtárában. A fájl hosszát és az eredeti állomány `md5sum` összegét összevetve ellenőrizd, hogy a másolás rendben megtörtént-e.

A következő lépés – ha van valamilyen fontos adatod a Zauruson – a biztonsági mentés készítése, ugyanis a ROM-frissítés során minden adat elvész. Most kapcsold be a Zaurust, és csatlakoztasd a hálózati adapterre. Bekapcsol a narancssárga, az akkumulátor töltését jelző LED. Nyisd fel az akkumulátor fedelét, de az akkumulátort ne vedd ki. Most jön a legnehezebb rész: miközben lenyomod a billentyűzet C és D gombját, nyomd meg egyszer a Reset gombot. Ez a gomb az akkumulátor fedele alatt található és tollal férhetsz hozzá. Talán egy segítővel könnyebb véghezvinni a gombnyomkodást. Ha sikerült, mindkét LED kigyullad, és megkezdődik a frissítési művelet. Ne nyúlj semmihez, a félbeszakadt frissítés kárt tehet a készülékben – a legjobb, ha elmész, iszol egy kávét, és utána visszajössz. Mindkét LED-et kikapcsolt állapotban fogod találni. Most már kiveheted a CF-kártyát, lezárhatod az akkumulátor fedelét, és megnyomhatod a Reset gombot.



Általában ott szokták a műveletet elrontani, hogy nem teljes ROM-lenyomatfájl kerül a CF-kártyára. Ismétlem, végezz ellenőrzést a másolás után, mielőtt behelyeznéd a kártyát a Zaurusba.

Adatkezelő alkalmazások

Adataidat lehetőleg ne a Zaurus GYK-ban leírt módszerrel próbáld áthozni. Adatbázist adatbázis után áttölteni, majd „átbitvadászni" a többi adatbázist – nem kellemes munka. Szerencsére a teljes áttérés egyetlen Perl-parancsfájllal is megoldható. Telepítsd a Perl 5.6.1, `opie-sh` és `p2z` csomagokat. Ezután másold az ***AddressDB.pdb***, ***DatebookDB.pdb***, ***MemoDB.pdb*** és ***ToDoDB.pdb*** fájlokat a Zaurusra, és futtasd le a `p2z`-t.
Most következhet a mobiltelefonod. Ha van rajta adatküldésre alkalmas infravörös kapu, a ***Zaurus Settings → Beam receive*** (Beállítások → Infravörös fogadás) menüpontjával indíthatod a vételt. A felbukkanó kérdéseknél az ***Add to Address book*** (Hozzáadás a címlistához) és az ***Add to Calendar*** (Hozzáadás a naptárhoz) parancsot választva adhatod hozzá az adatokat a megfelelő alkalmazáshoz.
A halovány személyi adatkezelő alkalmazások jelentik a Zaurus egyik legcsúnyább gyengeségét. Helyettesítésükre a theKompany.com tkcAgenda, tkcDatebook és tkcMemo alkalmazásait javaslom. A theKompany.com adatkezelő alkalmazásai tökéletes együttműködésre képesek az eredeti alkalmazásokkal, de ne feledd, hogy teljesen átveszik a helyüket, visszaállításra nem hagynak lehetőséget.
A theKompany.com részéről hasznos kiegészítő még a pénzügyeid követésére használható tkcKapital és a kiadásaid figyelemmel kísérésére alkalmas tkcExpense.

Összehangolás a Free Bonus Networking segítségével

A régebbi tenyérgépekkel ellentétben a Zaurus az adatokat IP-protokollal továbbítja az USB kapcsolat felett a gazdagép felé. Ha a gazdagépen maszkolást futtatsz, az adatok összehangolására használt kapcsolat általános hálózati kapcsolatba megy át. Az USB feletti hálózati hozzáférés beállításáról számos terjesztéshez létezik útmutató (lásd a ***Kapcsolódó címek*** részt). Ha sikerült összehozni, az adatok összehangolására a Sharp Qtopia Desktop vagy a Trolltech direct programját használhatod.

Irány az Internet!

Ha a Zaurus a dokkolóban van, USB feletti hálózati kapcsolatod is van. Ha csak vezetékes hálózatod van, nagyjából ötven dollárért szerezhetsz egy CompactFlash hálózati kártyát. Tagadhatatlan, a vezeték nélküli hálózat érdekesebb. A Zaurusszal használhatók közül a legjobb 802.11b kártya a Socket WL6000-320 jelzésű, alacsony fogyasztású kártya. Igaz, hogy a nagyobb kártyák – mint a Linksys WCG-11 – is működnek, de rengeteg energiát igényelnek. Más – például D-Link – kártyák használatakor nem lehet hozzáférni a hangcsatlakozóhoz és a tollhoz. Ha 1.1.x előtti ROM-ot használsz, bele kell nyúlnod néhány beállítófájlba. Mivel a korai Sharp ROM-ok biztonsági hiányosságokkal is terhesek, legjobb, ha a hálózat üzembe helyezése előtt frissítesz.
A Socket WL6000-320 valószínűleg hamarosan külön telepítés nélkül használható (Plug and Play) lesz, talán már akkor, amikor ezt a cikket olvasod. Nézd át a ROM-lenyomatodhoz tartozó leírást, és ha nem rendelkezik beépített támogatással, a ***Kapcsolódó címek*** segítségével rátalálhatsz a megfelelő illesztőprogramra.
Ha a Zaurus készen áll a 802.11b hálózatok kezelésére, szükséged lesz még egy hozzáférési pontra is. Ajánlom figyelmedbe a buherálásbarát Linksys WAP11-típust, ami a GNU/Linux SNMP megoldásaival nemcsak jól használható, de gépi szintű programja frissíthető, és irányított antennákkal kiegészítve nagy távolságú rádiós híd létesítésére is alkalmas.

Alkalmazások telepítése ipkg segítségével

A Linux alapú zsebgépekre szánt alkalmazások csomagolása már akkortájt megoldódott, amikor a Compaq iPAQ gépére felkerült az első Linux. Egy `ipkg` csomag alapvetően egy ***.tar.gz*** fájl, némi vezérlőadattal kiegészítve. Ha telepíteni akarsz egy alkalmazást, az alábbi paranccsal teheted meg:

```
# ipkg install csomag.ipk
```

Ellenőrizd, hogy ARM processzorra fordították-e le, és rendelkezel-e a szükséges könyvtárakkal. Ha el szeretnél távolítani egy csomagot, az `ipkg remove` parancsot kell kiadnod.

A lényeg: a játékok

Ha jó sok játékot akarsz a Zaurusodra, telepíts egy emulátort. Talán tetszenek a régi Nintendo Game Boy-játékok? Ha annak idején megvetted őket, csak a GnuBoyra lesz szükséged, no meg a ROM-fájlok jogtiszta áttöltését kell megoldanod.
A GnuBoy alkalmazást hozzárendelheted a ***.gb*** és ***.gbc*** fájlokhoz, így a játékokat egyszerűen, a megfelelő fájlokat a Documents fülről kiválasztva indíthatod el. A ***/home/QtPalmtop/etc/mime.types*** állományt az alábbi sorral kell bővítened:

```
application/gameboy gb gbc
```

A ***qtopiagnuboy.desktop*** állományhoz ezt fűzd hozzá:

```
MimeType=application/gameboy
```

1. kép A díjnyertes Froot
2. kép Az embeddedkonsole fülei olyanok, mint a KDE terminálja
3. kép drawpad – rajzprogram
4. kép Az IQNotes-szel a feljegyzéseket és a képeket tarthatjuk rendezetten
5. kép Gutenbrowser, az e-könyvek kezelője
6. kép Töltsd le most, olvasd később a pluckerrel és a Qt Readerrel

```
MimeTypeIcons=GnuBoy
```

A snes9x híres Nintendo-emulátor, a SuperNES-játékokat is kezeli. A natív játékok természetesen jobban és gyorsabban futnak, mint az emulált Game Boy-és SuperNES-programok. Biztosan ismered a Nethack nevű játékot – ugye jó volna, ha Zauruson is futna? Ha inkább sakkoznál, mivel a Zaurus Java-képességekkel is rendelkezik, szerezd be a Laser Chesst. Ne hagyd ki a Froot nevű, kirakószerű programot se, nemrég a Qtopia fejlesztői versenyen díjat nyert a Játékok kategóriában (lásd az *1. képet*).

MPEG- és Divx-filmek

A ➲ http://www.pocketmovies.net címről tölts le egy bemutatót, majd haladj az ***Applications*** → ***Media Player*** (Alkalmazások → Médialejátszó), ***Options*** → ***Full Screen*** (Beállítások → Teljes képernyő) sorrendben, csatlakoztass egy fülhallgatót, és élvezd a látványt. A készülékhez képest a minőség kiváló. 24 képkocka/másodperc frissítéssel le tudod játszani az MPEG-1 SIF (320×240) formátumú filmeket, de ha több filmet szeretnél tárolni, HALF SIF (160×120) filmeket is választhatsz. A ***Fit to Screen*** (Képernyő kitöltése) beállítást választva észre sem fogod venni az alacsonyabb felbontást.
Léteznek nagyobb tudású lejátszók is. Az opie-mediaplayer2 egyaránt támogatja az MP3, Shoutcast, Ogg Vorbis, Divx és egyéb formátumokat, de sajnos nagy mennyiségű memóriát igényel. Divx-filmekhez a tkcVideo a jobb, ez a folyamatos lejátszás és a jobb összehangolás érdekében a képkockák eldobására is képes.

Segédprogramok

A Tab Manager segít az indítómenü alkalmazásainak átrendezésében, ha pedig háttérfolyamatokat is futtatni akarsz, a Process Manager áll rendelkezésedre. A `kill` és `renice` lehetőségek nagyon jól jöhetnek démonok, konzolos alkalmazások vagy bizonytalanná vált folyamatok lekezelésekor.
Nem árt az SSH és egy jobb fajta terminál. A KDE konzolterminálja, az `embeddedkonsole-tabs` Qtopia alá is elérhető, a SourceForge oldalról tölthető le. Ha egyszerre több terminálra is szükséged van, lapokat is használhatsz benne (*2. kép*).
Ha már a SourceForge-oldalon jársz, szerezz be egy SSH-t is. A saját könyvtárad nem írható, ezért az SSH nem fog futni, ugyanis írni szeretne a ***/home*** könyvtárban található ***.ssh*** fájlba. Ezért írd át a ***/etc/passwd*** fájlt, és kezdőkönyvtárad ***/root*** helyett ***/home/root*** legyen. Ugyancsak erről a webhelyről szerezheted be az Emacs zaurusos változatát.
Ha 802.11b hálózatot is használsz, és az OpenSSH jelszó nélkül is fogadja a rendszergazdai kapcsolatokat, nem árt, ha jelszót adsz meg a gép védelmére. Ezt a ***Settings*** → ***Security*** (Beállítások → Biztonság) menüpontban teheted meg.
Ha a gépet a hálózat egyéb pontjairól, távolról is használni szeretnéd, telepítsd az fbvncserver csomagot. Ezután, ha a hálózat valamely pontján elindítasz egy VNC ügyfelet, máris láthatod a Zaurus kijelzőjének a tartalmát.
Ha képernyőképet szeretnél készíteni az éppen kipróbált alkalmazásról, a screenshot alkalmazás kézre esik. Terminálból kell indítani, az alábbi módon:

```
delay: sleep 2 ; scrshotcf
```

A Zaurus védelmét különféle, titkosítást is használó jelszókezelők segítségével erősítheted meg. Például a Keyring felülete barátságos, és `blowfish` algoritmust használ.
Ha STRIP alatt nagy mennyiségű adatot tárolsz PalmOS alapú gépen, jelszófájlodat a strip2zsafe segítségével alakíthatod át ZSafe formátumba. Bár a ZSafe felülete nem olyan szép, mint a Keyringé, nyílt szövegből történő importálási lehetősége miatt rendkívül könnyű áttérni rá. Hozz létre egy szöveges fájlt az alábbi formátummal:

```
"Kateg ria";"NØv";"FelhasznÆl nØv";"Jelsz ";
➥"MegjegyzØs"
```

Másold rá a Zaurusra, majd ZSafe alól válaszd ki; a tartalma – titkosítva – máris bekerül a jelszófájlba.

Térképek, rajzok, jegyzetek

Ha még bírod, néhány segédprogram és kiegészítő eszköz révén átérezheted azt a nyüzsgést, amely a Zaurus világát jellemzi. A qpeGPS segítségével a ➲ http://mapblast.com térképeit érheted el, illetve a útvonaladat követheted a tenyérgépen. Képnézegetőt sem csak egyfélét találsz. Figyelemre méltó a theKompany.com tkcGallery nevű programja és szabad egyenértékűje, a MooView.
A drawpad és az IQNotes (➲ http://iqnotes.kybu.sk) mindennapos segítőtársak. A drawpad (*3. ábra*) lehetővé teszi, hogy a Zaurus kijelzőjén gépelés helyett egyszerű vázlatokat készíts, az IQNotes (*4. ábra*) pedig hagyományos, többszintű elrendezésben jeleníti meg jegyzeteidet, rajzaidat.
Ha a bevásárlólista is rákerül, a Zaurus teljes értékű családtaggá válik. A Shopper segítségével ez is megoldható, így a tenyérgép érdekes játékszerből valódi segítővé válik. Az egyes boltokhoz külön listát készíthetsz, majd ahogy végigjárod őket, úgy jelenítheted meg a megfelelő listát is.

Nézz, nézz az ég felé…

Ugyan ki akarna azzal elrontani egy romantikus, holdfényes találkát, hogy nem ismeri a csillagokat? Töltsd le, és telepítsd a kicsit méretes ZaurusSkyExplorert. 4,5 MB-nyi adatbázisa segít megismerkedni a csillagokkal és égi alakzatokkal. Ha a műholdak helye is érdekel, a PetitTrackre és a NASA TLE állományaira lesz szükséged.

KAPCSOLÓDÓ CÍMEK

Kiegészítők és eszközök
Zaurus CF/SD eszközök listája, márkanév szerint rendezve
➲ http://enrage.dhs.org/~knight/ZaurusIndex.php
Socket WL6000-320802.11b kártya
➲ http://www.socketcom.com/product/WL6000-320.asp
Kis fogyasztású CF WLAN kártya üzembe helyezése Zaurus géppel ➲ http://www.cypherpunks.ca/zaurus/socket.html
Fém és bőr hordtok Zaurus SL5500-hoz, Portfolio modell, Limited Edition ➲ http://www.extreme-limit.co.jp/english/mobilepc/zaurus_sl5500_port_lim.htm
Sumdex GLP-824 bőrtok ➲ http://sumdex.com/glp-824.htm
Linksys WAP11
➲ http://linksys.com/products/product.asp?prid=157

Buherálás
3,5 mm-es csatlakozóba épített apró mikrofon
➲ http://www.rundel-d.com/palmtop/zaurus/microphone.htm
Mikrofon beültetése SL-5x00 készülékbe
➲ http://www.opie.us/opie/devZone.php/sl5x00_mic.htm

ROM-képek
Sharp, letöltések
➲ http://more.sbc.co.jp/slj/download.asp
Opie (Open Palmtop Integrated Environment)
➲ http://opie.us
Paul ➲ http://www.btinternet.com/~p.flinders/sl-5000d
Crow ➲ http://www.schwag.org/~crow
modzaurus (saját ROM-képek készítéséhez)
➲ http://www.ece.uah.edu/~wes

Segédprogramok és fejlesztői eszközök
Perl
➲ http://www.frontgarden.net/~matthew/technical/zaurus/perl.html
opie-sh ➲ http://www.rit.edu/~tfs1812/feed
p2z (Palm–Zaurus átalakító eszköz)
➲ http://mobilix.org/p2z.html
USB feletti ethernetkapcsolat beállítása
➲ http://www.ruault.com/Zaurus/ethernet-over-usb-howto.html
Tab Manager
➲ http://www.csh.rit.edu/~benjamin/desktop/zaurus/tabmanager.shtml
Process Manager
➲ http://www.killefiz.de/zaurus/showdetail.php?app=320
embeddedkonsole-tabs, SSH, Emacs és screenshot kisalkalmazás
➲ http://prdownloads.sourceforge.net/zaurus
fbvncserver ➲]http://OpenSource.lineo.com/fbvncserver
Keyring ➲ http://www.scrypt.net/~celer/kweb
Zaurus Password Manager, ZSafe
➲ http://home.t-online.de/home/CarstenSchneider/zsafe
strip2zsafe ➲ http://mobilix.org/strip2zsafe.html

Térképek, rajzok, jegyzetek
qpeGPS ➲ http://qpegps.sourceforge.net/gpegps.html
MooView ➲ http://www.cs.unc.edu/~scheuerm/mooview
drawpad ➲ http://www.zaurusoft.com/packages/view/89
IQNotes ➲ http://iqnotes.kybu.sk
Shopper – „The Shopping List Manager for the Zaurus”
➲ http://www.blackie.dk/Zaurus/Shopper

Nézz, nézz az ég felé…
ZaurusSkyExplorer
➲ http://members.aon.at/ritchie/de/index.html
PetitTrack ➲ http://www.qsl.net/n1vtn/petittrack.html

Hangok és elektronikus könyvek
CMU Speech Tools
➲ http://www.speech.cs.cmu.edu/flite/packed/flite-1.1
Gutenbrowser
➲ http://sourceforge.net/projects/gutenbrowser
QT Reader ➲ http://www.timwentford.uklinux.net
Plucker ➲ http://www.plkr.org

Személyi adatkezelő alkalmazások
tkcAgenda, tkcDatebook és tkcMemo
➲ http://www.thekompany.com

Játékok
GnuBoy ➲ http://www.warmi.net/zaurus/gnuboy.shtml
snes9x
➲ http://sourceforge.net/project/showfiles.php?group_id=39939&release_id=70845
NetHack ➲ http://trolls.troll.no/warwick/nethack
Laser Chess Java
➲ http://www.laserchess.org/downloads/laserchess_1.09_arm.ipk
Froot ➲]http://www.cs.unc.edu/~scheuerm/froot

Szórakozás
tkcvideo
➲ http://www.thekompany.com/embedded/tkcvideo
opie-mediaplayer2 ➲ http://arachni.kiwi.uni-hamburg.de/~harlekin/op2

Internet és kapcsolattartás
tkcJabber ➲ http://www.thekompany.com
QtJim ➲ http://www.telecom.tuc.gr/~perak/utils/qtjim
KinkattaLite
➲ http://www.csh.rit.edu/~benjamin/desktop/zaurus/kinkattalite.php
KMerlin ➲ http://kmerlin.olsd.de
Zic (Zaurus Internet Chat) ➲ http://zic.sourceforge.net
Kismet ➲ http://kismetwireless.net
kismet-qte ➲ http://sourceforge.net/projects/kismet-qte
the Familiar Project ➲ http://familiar.handhelds.org

Hangok és elektronikus könyvek

Angol nyelvű CMU beszédeszközök immár Zaurusra is elérhetők. Bármely szöveges állományt fel tudnak olvasni. A brit angol ugyan elég erős akcentussal szólal meg, de más tenyérgép-tulajdonosok elkápráztatására vagy éppen a napi hírek átfutására mindenképpen jól jöhet. Csak remélhetjük, hogy hamarosan az elektronikus könyvkezelő alkalmazásokba is beépítik. A Gutenbrowser közvetlenül a Project Gutenberg anyagából

keres klasszikus irodalmi műveket, és egyenesen a kezedbe tölti le őket (*5. ábra*). Ügyes kiegészítő a QT Reader, amely az egyszerű szöveges fájlokkal, a ***.pdb***-dokumentumokkal és a Plucker formátummal egyaránt boldogul (*6. ábra*). Az utóbbinak köszönhetően lehetőséged nyílik arra, hogy weboldalakat tölts le a tenyérgépre, majd később, akár hálózati kapcsolat nélkül olvasgasd őket. Szabadon választhatsz, hogy híreket, vicceket, vagy egyéb tartalmakat viszel magaddal.
A Plucker teljesen olyan, mint az AvantGo, neked csak a megfelelő címet kell ismerned. A legnehezebb része a dolognak az, hogy kifejezetten tenyérgépekre készült tartalmat találj, ugyanis az AvantGo szerződés a tartalomszolgáltatóknak megtiltja, hogy tenyérgépre készített anyagaik címét közzétegyék, vagy más kapcsolat nélküli olvasóprogramoknak is átadják.
Persze némi keresgéléssel azért találhatsz tartalmat, ilyen például az ausztrál The Age; valahol a főoldalukon rálelhetsz a tenyérgépekre készült változatra mutató hivatkozásra.
Gyűjtsd össze a címeket, majd a ***Spider.py***, a Python alapú Plucker-böngésző segítségével rendezheted őket. A létrejött ***.pdb*** fájl tartalmát később bármikor elolvashatod.

Matek

Noha a Zaurus gyárilag is elég jó számológéppel rendelkezik, a programozható függvények hiánya miatt például elég nehézkes vele a mennyiségek átváltása. A ZUC ellenben remekül megfelel erre a célra. Két zárt fejlesztésű matematikai eszköz a NeoCal és a Formulae 1. A NeoCal jelenleg a legjobb ilyen program Zaurusra, rengeteg pénzügyi, tudományos és statisztikai függvényt ismer. A Formulae 1 egyenletek megoldására használható, érdekessége, hogy nemcsak a végeredményt, de a köztes lépéseket is megjeleníti. Ha ilyesmire van szükséged, érdemes megvenni.

Fájlok, könyvtárak és parancsfájlok

Mielőtt hálózati alkalmazásokkal kezdenél foglalkozni, tudnod kell, hogy a Sharp fájlrendszerében alapállapot szerint hol találhatók:

- ***/home/root/usr/lib/ipkg***: az egyes alkalmazások által telepített fájlok listája;
- ***/home/QtPalmtop/apps***: az indítómenüben szereplő elemek leírófájljai;
- ***/home/QtPalmtop/bin***: futtatható állományok;
- ***/home/QtPalmtop/pics***: PNG formátumú ikonok és egyéb képek;
- ***/home/root/Applications***: az alkalmazások adatállományai (jellemzően XML formátumban);
- ***/home/root/Settings***: bizonyos alkalmazások beállítóállományai.

Ha egy memóriakártyára alkalmazások vannak telepítve, létrejön rajta egy QtPalmtop könyvtár, így tudod módosítani az alkalmazások megjelenését.
A rendszerindításkor indítandó programokat egy ***/etc/rc.d/rc5.d*** fájlba kell telepíteni. A fájl neve legyen például ***S99local***. Az alábbi példában a környezeti változókat és az állomásnevet adjuk meg:

```
#! /bin/sh
#
# S99local    run local startup stuff
#
PATH=/bin:/usr/bin:/sbin:/usr/sbin
LD_LIBRARY_PATH=/lib:/usr/lib:/home
->QtPalmtop/lib

export LD_LIBRARY_PATH PATH

hostname sajat_allomas.sajat.tartomany.hu
```

Az `S99local` legyen futtatható:

```
chmod 750 S99local
```

Internetes alkalmazások

Azonnali üzenetküldésre kiváló választás a theKompany.com tkcJabber programja. Segítségével – ha a háttérben futtatod, illetve ha a Zaurus 802.11b kapcsolattal rendelkezik – a készülék azonnali zseb-üzenetküldő eszközzé válik. Az ICQ, AIM, MSN, Yahoo Messenger vagy IRC használata pontosan olyan lesz, mintha mobiltelefont vinnél magaddal. Ha üzeneted érkezik, a Zaurus bejövő üzeneteket jelző LED-je villogni kezd, és mobiltelefonos jellegű hangjelzés szólal meg. A csevegést külön ablakban folytathatod.
A dolognak két apró szépséghibája van. Először is, kezelői felületén kizárólag ikonok vannak. A legnehezebb az első használat lesz, hiszen egy árva tippet, feliratot sem találsz sehol. Szerencsére a felület rendkívül ötletes, működésére – nagyrészt – hamar ráérzel. Nekem még rá kell jönnöm, hogyan lehet ICQ-azonosító vagy becenév alapján felvenni valakit az ismerősök listájába. A másik gond a tkcJabber IRC támogatása. Ha rákattintasz valamelyik rejtélyes ikonra, IRC-kapcsolatod megszakad. Ezek a hiányosságok elviselhetők, a program mindennapos használatát lényegesen nem befolyásolják.
Léteznek ingyenes azonnali üzenetküldő programok is: a QTJim Jabber, a KinkattaLite AIM, a KMerlin pedig MSN hálózatra tud kapcsolódni. IRC-zni a Zic a legjobb, de Yahoo Messenger vagy ICQ ügyfelet ingyeneset nem találsz, tehát mégiscsak maradjunk annál, hogy legjobb a tkcJabber.
Hm, esetleg egy kis szimatolás a 802.11b hálózaton? A Kismet remekül megfelel e célra. Ha `prism2` alapú vezeték nélküli hálózati kártyád van – ilyen például a Linksys WCF-11 –, akkor nem lesz gond a telepítésével. Socket kártyával is működő változatot a Socket illesztőprogram lelőhelyéről szerezhetsz be. A `kismet-qte` kiváló grafikus felületet ad a programnak. Hálózati hibaelhárításra az `nmap` áll rendelkezésedre.

Összegzés

Remélem, sikerült felhívnom a figyelmedet néhány nagyszerű, Zaurusra készült programra. A legjobb az, hogy az alkalmazások többsége GPL szerződéssel jelenik meg. Ha – hozzám hasonlóan – korábban Palm Pilotod volt, és szükséged van néhány különleges programra, nos, neked egy Zaurus kell.

További kapcsolódó címek a 44. CD Magazin/Zaurus könyvtárában találhatóak.

Linux Journal 2003. január, 105. szám

Guylhem Aznar
A The LDP (➲ http://www.tldp.org) irányítója, a GNU Projekt dokumentációs vezetője. Noha hatodéves orvostanhallgató, jelenleg informatikából írja doktori dolgozatát, kevéske maradék idejében pedig Zaurusával játszadozik. Az ➲ http://externe.net oldalról érhető el.

Keleten a helyzet...

Ül az amerikai apuka a számítógép előtt, és bosszankodik. A lánya zenét töltött le, de a saját gépére mentette, így apuka azt most nem tudja meghallgatni.

Amerikai apuka ekkor kitalálja, hogy elmegy a boltba, levesz a polcról egy kisebb kiszolgálót, és a másnapi napiparancsban kihirdeti a család számára: ezentúl mindent az új jövevényre kell menteni. Amerikai apuka és családja Linux-felhasználók lettek, még ha nem is tudnak róla. Valószínűleg a tengerentúli valóság sem teljesen ilyen, de a magyarral nem indíthattam, az ugyanis elég lehangoló. Bár már vannak olyan családok, ahol nemcsak egy pályázati úton szerzett számítógép lapul szerényen a sarokban, hanem több gépet, esetleg családi kiszolgálót is találni, ezek egyelőre nagyon ritkák.

A magyar ugaron

Talán ez is az oka annak, és a szokásos „barkácsoljunk" magyar virtus, hogy a Demand 2000 munkatársai a gyerekkori sárkányos mesék világát idéző elkeredő szemmel mesélték, amit talán azóta is próbálnak felfogni: már két darabot is eladtak a Toshiba otthonra és kisebb irodákba szánt kiszolgálójából.
Az ütött-kopott, néhol kicsit horpadozott doboz láthatóan nem a szomszéd utcából érkezett. A Toshiba America feliratok, az amerikai szabvány szerinti tápcsatlakozó – ezt az importőr egy magyarral egészítette ki –, a kizárólag angol nyelvű leírás mind-mind arra utalt, hogy a magyarországi eladások nagyságrendje miatt még jó ideig nem leszünk kiemelt célterület a távol-keleti cég számára.
A készüléket úgy csomagolták, ahogy a hordozható gépeket szokás: oldalt egy laposabb dobozban külön vannak a kellékek, a gépet magát pedig középen, két szivacstartóval biztosítva helyezték el.

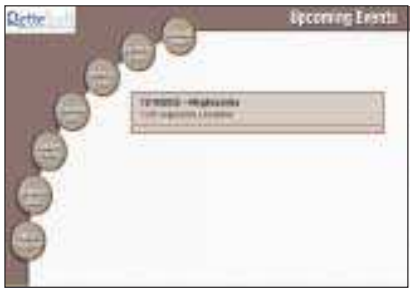
A szép külső nem minden...

A nagy Ő szép. Szürke fémháza, egyszerűségében elegáns előlapján fénylő Toshiba felirata, a sarkaira húzott acélkék műanyag betétek legalább egy hosszú sóhajt megérdemelnek – eddig mindenki beleszeretett, akinek megmutattam. Ha lakásunk nem szigorúan a neobarokk stílust követi, bátran kitehetjük akár a könyvespolcra is. Nagyon nem szabad elrejteni, hiszen a műszaki érdeklődésűek arca éppen a készülék hátlapjának láttán fog kipirulni.
Hátul, a tápcsatlakozó mellett egy RJ-45-ös aljzatot találunk, majd még

nyolcat, ezen kívül egy telefonos aljzatot és egy párhuzamos kaput is. Két bumszli csavarral egy kisebb fedlapot is kioldhatunk, alatta egy 20 GB-os IBM Travelstar merevlemez lapul. Mellé egy másikat is telepíthetünk, majd szükség szerint tükrözhetjük a két lemezt, így készítve egyfajta biztonsági mentést, vagy dönthetünk úgy is, hogy további tárhelyet szeretnénk, a tükrözésről pedig lemondunk.
A bővítési lehetőségek rövidke sora ezzel véget is ér. A készülékre monitort, billentyűzetet, egeret nem lehet csatlakoztatni. Tervezése során nyilván nem az volt a cél, hogy a felhasználó a gépet piszkálja, hanem az, hogy használja. Ehhez minden segítséget meg is kap: a 414 oldalas használati útmutató a félreeső beállítási lehetőségeket is részletesen tárgyalja.
Azt, hogy miért kell nekem mindenképpen a telepítő CD-lemezt használnom, már kevésbé részletesen, ám annál hangsúlyosabban fejtették ki. Mivel nem tudtam, mibe tenyerelek, az előírt módszertől nem is mertem eltérni. Az alapelképzelés – minő előítéletek! – az, hogy a felhasználó gépén Windows fut. Ennek beállításait a telepítő így-úgy módosítja, és újraindítás után a kiszolgáló már használható is. A telepítőprogram nem sokat kérdezett, csak egy felhasználónevet kért. Azt ugyan elfelejtette pontosítani, hogy miféle felhasználónévre kíváncsi, így a megadott névvel örömömre új felhasználót hozott létre a Windows alatt és a kiszolgálón egyaránt, kirakott az asztalra három ikont, aztán újraindította a gépet.
A kiszolgálót addig naná, hogy nem találta meg, míg belső hálózati ethernetkártyámat DHCP-használatra nem állítottam. Innen viszont gördülékenyen mentek a dolgok. Saját gépem DHCP-n keresztül kapott címet a kiszolgálótól, amely az alapértelmezett átjáró szerepét is átvette.
A kezdeti szenvedés után innentől felgyorsultak a dolgok. A már említett ikonok segítségével a kiszolgálón három fontos felületet érhetünk el webalapon. Az egyik a felügyeleti program, a másik a digitális központ, amely zenék, filmek, képek tárolására és megosztására használható, a harmadik pedig a belső csoportmunkaportál.
A felügyeleti programról csak jót tudok írni. Ízléses, jól áttekinthető, ha valamelyik elem fölé odahúzzuk az egeret, rövid súgót jelenít meg. Egy webes felügyeleti felület tervezésekor bizonyára sok fejtörést okoz, hogy milyen szintű felhasználó számára készüljön a termék: a kezdőknek nem szabad túl sok beállítási lehetőséget adni, mert valamit biztosan el fognak rontani, a profik viszont elégedetlenek lesznek, ha nem engedik őket mindenbe belepiszkálni. Úgy vélem, a Toshiba tervezői csaknem tökéletesen eltalálták a középutat, és általános környezetben elegendő beállítási lehetőséget adnak.

A rendszergazdai felület

A rendszergazdai felület öt csoportba sorolja a beállításokat. A ***Report*** csoport tagjaival kérdezhetjük le a készülék állapotát és beállításait, a tűzfal állapotától kezdve a processzor tápfeszültségén keresztül a legutóbbi biztonsági mentés időpontjáig. A ***System*** csoport teszi lehetővé többek közt a felhaszná-



lók hozzáadását, az elektronikus postaládák beállítását, a nyomtatási sor áttekintését és a biztonsági mentések és visszaállítások elvégzését. A ***Network*** csoport neve beszédes, itt adhatjuk meg internetkapcsolatunk jellegét (ethernet alapú, kábelmodem, DSL); állíthatjuk be az internetes tartalomszűrést; a tűzfalat, amelyhez saját kezűleg is hozzáadhatunk szűrési szabályokat; a kiszolgáló helyi hálózati nevét és munkacsoportját; illetve VPN-kapcsolat létrehozására is módunk van.
A belső csoportmunkaportál felügyeletére szolgál az ***Intranet*** csoport. Az egyszerű portál kinézetét néhány egyszerűbb sablon közül választhatjuk ki, egyedivé tételére megadhatjuk a céges – avagy családi – logót. A portálka tartalmaz egy üdvözlő oldalt; egy híroldalt, amelyhez egyszerű szerkesztői háttérrendszert is készítettek; egy eseményoldalt, amelyre a többeket is érintő eseményeket, találkozókat jegyezhetjük fel; egy dokumentumtárat, ezt a közös sablonok elérhetővé tételére használhatjuk; valamint egy webes hivatkozásokat tartalmazó oldalt.
A *Services* csoport a leghaszontalanabb számunkra: itt tudjuk bejegyeztetni a készüléket, támogatást kérhetünk, frissítéseket tölthetünk le – ezek önmagukban hasznos dolgok, ám innen, Európából elég bajosnak tűnik az igénybevételük.
A kiszolgálón Samba fut, alatta minden felhasználó egy nyilvános és egy személyes könyvtárat kap, amelyekhez a telepítőprogram egy-egy hálózati meghajtót is hozzárendel. A felhasználókat a rendszer három jogosultsági szintre osztja, az első szint például nem érheti el a felügyeleti felületet. Ez kiegészül azzal, hogy például VPN-kapcsolatnál külön jelölhetjük ki azokat a felhasználókat, akiknek joguk van a kapcsolat használatára.
Természetesen a felügyeleti felület nem kínál lehetőséget az összes beállítás módosítására. Nem sikerült például megtalálni a módját annak, hogy módosíthassam a belső hálózati ethernetkártya IP-címét, vagy éppen a DHCP-kiszolgáló által osztogatott címtartományt. Valószínűleg ezek azok a beállítások, amelyek a célközönség számára nem fontosak – gondolhatták a tervezők.

Hatalomátvétel

Természetesen nincs minden veszve, `telnet`-tel ugyanis be tudunk jutni a gépre. Erre egyetlen felhasználónak van joga, őt ***telnetuser***-nek hívják. Miután ekként beléptünk, az alábbiak jelennek meg:

```
Red Hat Linux release 6.1
(Cartman)
Kernel 2.2.12-20 on an i586
```

Ha a `su root` parancson is túljutnunk szabadon garázdálkodhatunk: a grafikus felületről hiányzó beállításokat is módosítani tudjuk, illetve alkalmazásokat telepíthetünk; egyedül arra kell vigyázni, hogy egy elrontott frissítés vagy egyéb hiba miatt – billentyűzet és egér hiányában – nehogy kizárjuk magunkat a saját gépünkről.
Mire használható egy ilyen viszonylag zárt kiszolgáló? Ha valakinek fel kell tennie ezt a kérdést, akkor a válasz: semmire. Ha sikerül felismerni a benne rejlő lehetőségeket, akkor az, hogy gyorsan és kényelmesen helyezhetünk üzembe egy otthoni vagy kisebb irodai kiszolgálót.

A készüléket a Demand 2000 Kft. munkatársai bocsátották rendelkezésemre. Végfelhasználói ára: 89 000 Ft + áfa.

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.

A gép adatai

AMD K6-2 350 MHz processzor
64 MB SDRAM
20 GB merevlemez
telefonos csatlakozó
hétkapus 10/100 Mb/s ethernetkapcsoló
ethernetaljzat kábelmodem/DSL csatlakozáshoz
párhuzamos kapu
előlapi LCD
2 kg
6,84×34×26 cm

Képességei

- fájlkiszolgáló
- nyomtatókiszolgáló
- internetmegosztás
- helyi hálózati portál
- tűzfal
- webes gyorsítótár
- biztonsági mentés

Tanuljuk meg a MySQL használatát 24 óra alatt

Amikor könyvet mutatunk be, szerintem nagyon fontos tisztázni, hogy az adott könyv kinek is szól. Aki ismeri már a ***24 óra*** sorozatot (mondjuk a PHP4-es vagy a JavaScriptes kötetet), tudja, hogy a sorozat célja egy-egy téma bemutatása, általában a kezdő szinttől indulva, de gyors lépésekkel haladva. Természetesen ebből jó és rossz tulajdonság is következik. Az első tulajdonság, hogy ugyan az alapoktól indul, nem ragad le ezen a szinten, a második viszont, hogy mivel az alapoktól indul, nem marad sok hely egy-egy kényesebb kérdés részletes kivesézésére.

A MySQL-könyv a maga 400 oldalával pontosan beleillik a sorozatba: olyanoknak íródott, akik értenek valamennyit a programozáshoz, de SQL alapú adatbázis-kezeléssel nem feltétlenül foglalkoztak. Az első 40 oldal gyors „alaptanfolyamnak" tekinthető, amelyben az újoncok a relációs adatmodellekkel ismerkedhetnek meg, sőt még a normalizálással is. Ez a rész rendkívül rövid, de egy gyors felfogású embernek elég. Természetesen nem veheti fel a versenyt egy Halassy-könyvvel, de nem is ez a célja.

A második rész mintegy 30 oldal terjedelemben foglalkozik a környezet beállításával, itt kell megemlítsem a könyv egyik érdekes elméletét. Annak ellenére, hogy a windowsos és linuxos környezettel is foglalkozik, egyértelműen azt a vonalat képviseli, miszerint a kiszolgáló Linuxon fusson, a munkakörnyezet viszont Windows legyen. Ez a szerző egyéni döntése, ebben a kérdésben nem értek vele egyet, hiszen remek eszközök vannak már Linux alá is, így nyugodtan alakíthatunk ki teljesen a linuxos fejlesztőkörnyezetet is. Erre a kérdésre a szerző egyébként a könyvben később még visszatér, amikor részletesen beszél arról, hogyan hozzunk át adatokat Accessből, de például a más adatbázisokkal tartott kapcsolatoknak csak a „szöveges export" változatát mutatja be.

A harmadik résznek különösképpen örülök, a könyv ugyanis rászoktat, hogy ne úgy essünk neki az adatbázis létrehozásának, mint tót az anyjának, hanem szánjunk időt a táblaszerkezet átgondolására, a táblák létrehozására, a megszorítások és jogosultságok kialakítására. Igaz, a CREATE paranccsal egy kicsit tovább szöszöl, mint amennyit szerintem szükséges.

A könyv írójának szinte egy teljes SQL-alaptanfolyamot sikerült belesűrítenie a negyedik és ötödik rész 110 oldalába.

Tanuljuk meg a MySQL használatát 24 óra alatt

Kiadó: Kiskapu Kiadó
Eladási ár: 4200 Forint
ISBN: 963 9301 49 3
A könyv honlapja:
➲ http://www.kiskapu.hu

Remekül felépített részről van szó, mely megismertet az SQL két leggyakrabban használt, és minden kétséget kizáróan legtöbb lehetőséget rejtő utasításával, a SELECT-tel és a DELETE-tel. Emellett két fejezetbe összegyűjtve végigfut a beépített függvények három fontos csoportján: a szövegkezelő, a numerikus és a dátumkezelő függvényeken. Sajnos, a függvényekre szánt 20 oldal rendkívül rövid, és nem is elég arra, hogy a lelkivilágukba beleássa magát az ember (főleg, ha több típusú adatbázissal kell dolgoznia, és szembekerül a dátumformátumok kesze-kusza világával), de elégséges ismertető ahhoz, hogy egy referencia alapján vígan eldolgozgasson.

Lényegében a hatodik, a tranzakciók használatáról szóló rész az „alaptanfolyam" befejező szintje, ami viszont annyira fontos téma, hogy a szerző kettéválasztotta. Talán erre a területre igaz különösképpen a mondás, hogy gyakorlat teszi a mestert. Aki komoly munkát akar végezni SQL alapon, készüljön fel rá, hogy nagyon sok gyakorlatot igényel a tranzakció-kezelés megismerése. Ezért nem is csoda, ha azt mondom, az a 40 oldal csupán elméleti alapnak tekinthető, rengeteg gyakorlás kell mellé!

A rendszerfelügyelettel foglalkozó hetedik rész meglepő egy ilyen könyvben, hiszen bevezető jellegű könyvekben általában csak a program használatával foglalkozunk. Ez a rész olyan feladatokkal ismertet meg minket, mint az adatbázis mentése, visszaállítása, a felügyeleti parancsok vagy a finomhangolás. A könyv nagy előnyének tartom, hogy valóban felkészíti az olvasót az életközeli feladatokra, már amennyire ez a terjedelembe belefér. Hiába tudunk remek lekérdezéseket készíteni, ha egy költözésnél nem tudjuk, hova kapjunk, vagy ha éles rendszeren nem tudunk biztonsági mentést készíteni.

El is érkeztünk az utolsó részhez, amelyben a szerző programnyelvekből használja a kiszolgálót. Lényegében két nyelvet említ komolyabb szinten, a Perlt, és a PHP-t. Először bosszantó volt, hogy nem mutat be olyan teljes megoldás, mint például testvére, a ***Tanuljuk meg a PHP4 használatát 24 óra alatt***, elvégre ez tenné fel az i-re a pontot. Ugyanakkor el kell fogadjam, hogy a könyv így is sokkal többet adott, mint amennyit első pillantásra kinéz belőle az ember, és ez a téma tényleg olyan komoly, hogy nem lehet tíz-húsz oldalban „lezavarni". A könyv még egy függeléket is tartalmaz, amiben hasznos dolgokat találunk (telepítés, lefoglalt szavak, MyODBC használata, példaprogramok), és további ötleteket és példákat kapunk például a PHP használatára.

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