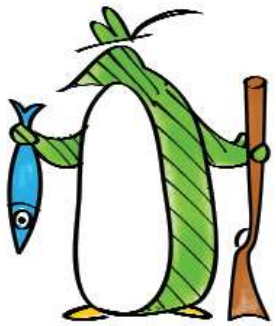
Beköszöntő

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

Kis hazánkban is egyre népszerűbb a Linux, amit nagymértékben a SuSE-, illetve az UHU-Linux-terjesztéseknek köszönhet. Egyre jobb a magyar nyelvű támogató, súgó és egyéb felhasználókat segítő rendszer, így a kezdő Linux-felhasználók is nagyobb biztonsággal kezdhetnek neki feladataik vagy felmerülő nehézségeik megoldásához. A felhasználói felületek és programok magyar nyelven (többé-kevésbé) szólnak a felhasználókhoz, ezzel is könnyítve a tájékozódást. A Linux-felhasználók dolgát nehezíti viszont a kiadások változatossága, hatalmas különbségek vannak változat és változat között, amíg az egyik rendelkezik egy szinte mindenre kiterjedő grafikus beállítóeszközzel, a másiknál még mindig csak a „fapados, turkálok a beállításfájlban”-megoldás marad. A kezdőknek nehéz megtalálni az igazán megfelelő Linux-változatot. Ezen az úton segíthet elindulni a fentebb említett két terjesztés. Az UHU-Linux újabb mérföldkőhöz érkezett, megjelent a Kamion fedőnevet viselő 1.1.1-es változat, egyre többen használják és szeretik a magyar linuxosok közül.
A kezdők számára tényleg kiváló rendszer, ezért a CD-mellékletünkön is megjelentetjük, ezzel is elősegítve a terjedését. A telepítő nagyon egyszerű, nem lehet gond senki számára sem az alaprendszer telepítése és beállítása.
A vezérfonalunkban felölelt téma a programfejlesztőknek nyújthat felüdülést. A Xnee programmal egy fáradhatatlan fejlesztőtársra tehetünk szert: helyettünk próbálgatja a fejlesztett programunkat. Ezenkívül tanítható is, ilyen módon bármilyen feladatra fel lehet készíteni.
Minden kedves olvasónak hasznos időtöltést kívánok lapszámunkhoz!

Programvadászat



Amikor a Kamion elgázol

Az UHU-Linux 1.1-es változata követte a szokatlan névválasztási szokást, így teremtőitől a Kamion nevet kapta. Sok tekintetben különbözik az 1.0-s változattól, használhatóságában és minőségében is hatalmas lépést tett az elődhöz képest.

A telepítés

A telepítés, ha lehet még egyszerűbb, mint eddig, nem számoltam meg, de néhány kattintás és már a gépemen is csücsült az újdonsült UHU-Linux. Ez időben ugyan nem volt kevés, mivel 2,1 GB adatot kellett a 650 MB-os CD-ből kicsomagolni és telepíteni. Szerencsére nem tapasztaltam a képernyő rossz kezelését a telepítés alatt, ami az előző változatnál nagyon zavaró volt, s az X kipróbálása sem jelentett örök halált a gépnek.

A rendszer

Ha választanom kellene a Linux-változatok között, hogy a kezdők számára mit ajánljak, azt hiszem, ebben a versenyben jó eséllyel indulna az UHU. Nagyon tetszett, hogy az előző változathoz képest a mimetype-okat kiválóan kezeli a rendszer. Csak kattintanom kellett a dokumentumok ikonjain és máris megnyílt a megfelelő alkalmazással. Egy másik operációs rendszer alatt a felhasználók már régóta használhatják azt a szolgáltatást, hogy egy tömörített állományból a grafikus fájlkezelő segítségével megnyithatják a fájlokat minden csűrés-csavarás nélkül. Kipróbáltam, hátha működik itt is, de sajnos nem sikerült működésre bírnom. Ugyan felismerte a fájl típusát, de nem tudta megnyitni, mindig a */tmp/fájlnévre* hivatkozott, hogy ott bizony nincsen ilyen fájl.
A másik érdekesség a KDE/Gnome közötti váltásban tapasztalható: egyszercsak lett egy *Szemétkosár* és *Kuka* ikonom, ezen felbuzdulva kidobtam a *Szemétkosarat* a *Kukába,* így helyreállt a rend az asztalomon.
A programok összeállítása célravezető, kedvünkre találhatunk programot szinte minden feladatra.

A változások a teljesség igénye nélkül

- A telepítőbe bekerült az `agp` magmodul, így bizonyos videokártyák esetén kipróbálás közben már nem száll el a grafikus felület.
- A fölösleges és egyben hibásnak tűnő `pdc-ultra` modulra történő utalást kiszedték a PCI adatbázisból.
- Nagyon sokat fejlesztettek az UHU-vezérlőpulton.
- A Gnome panelén lévő óra átméretezi magát, ha a megjelenítendő szöveg nem férne ki (például szerdáról csütörtökre váltáskor).
- Az XMMS programban szünet (pause) állapotból bizonyos hangkártyák esetén nem lehetett folytatni a lejátszást.
- Az Audacity programot az 1.2.0-pre4 változatról a végleges 1.2.0-ra frissítették, és tökéletesítettek a magyar fordításon.
- A `synaptic` csomag indulásakor esetenként megjelenő figyelmeztető ablakban kijavították az ékezethibákat.
- Az `mplayerplug-in` csomagot újabb, lényegesen jobban működő változatra frissítették.
- Apró hibajavítások jelentek meg a `devfsd` és az `uhubuild` csomagban is.

A már telepített UHU-Linux 1.1 rendszer esetén a csomagokat az `apt-get update`, majd `apt-get upgrade` paranccsal tudjuk frissíteni. Modemes felhasználók, a hálózati forgalom igen jelentős csökkentése érdekében ne az alapértelmezett FTP protokollt használják, hanem az `rsync`-et. Ilyenkor az `apt-get` nem a teljes új csomagot tölti le, hanem a régi változat ismeretében csak a régi és az új változat közötti különbséget. Az `rsync`-et használó frissítés megkezdése előtt a részletekért mindenképpen figyelmesen olvassuk el a */usr/share/doc/Packages/apt/RSYNC_HU* fájlban található leírást.
A két új ISO image viszonylag kevés helyen különbözik az egy hete kiadott változattól. Ezért a letöltéshez itt is ajánljuk az `rsync` protokoll használatát. Előkészületként a már letöltött ISO fájl(oka)t valamilyen Linux fájlrendszerű lemezrészre kell másolnunk, a VFAT fájlrendszer nem megfelelő. Ezen a lemezrészen szükség van további legalább egy CD-nyi üres helyre. Ha ezzel megvagyunk, a következő paranccsal írhatjuk felül a régi ISO fájlunkat az új változattal:

```
rsync -av --progress
rsync://rsync.uhulinux.hu/ftp/uh
u/1.1/uhu-linux-1.1.1-cd1.iso
uhu-linux-1.1-cd1.iso
```

Az utolsó tulajdonság (argumentum) a már létező helyi fájl, vagyis a régi ISO image neve, szükség esetén útvonallal együtt. A második CD letöltése ugyanilyen paranccsal történhet, `cd1` de helyébe `cd2`-t kell írnunk.

Jó szórakozást mindenkinek! Vagyis "uhu-ul": Juhhú!

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

Megtörtek

A Wind River, a VxWorks gyártója bejelentette, hogy kínálatát Linux alapú megoldásokkal bővíti. A valós idejű operációs rendszerek terén

vezető szerepet játszó cég részéről mindez hatalmas előrelépést jelent, hiszen a Wind River mindeddig erősen ellenezte a nyílt forrású fejlesztéseket. Ezzel egy időben megjelent a Linuxszal szorosan együttműködni képes VxWorks 6.0 operációs rendszer, valamint a cég új Wind Power 2 integrált fejlesztői környezete is. További újdonság, hogy a VxWorks ügyfelei a továbbiakban a korábbinál kedvezőbb díjfizetési módokat vehetnek igénybe. A Wind River bevétele az utóbbi években folyamatosan csökkent, amiben elsősorban a Microsoft beágyazott Windows-változatainak és a MontaVista Linuxnak a terjedése játszott szerepet. A cég részéről tehát nem megvilágosodásról, inkább a túlélés érdekében szükségesnek vélt lépésről van szó – azt, hogy ez mennyire lesz eredményes, néhány év múlva megtudjuk.

➲ http://www.windriver.com

Kézitévé

A Digital Video Broadcast (digitális műsorszórás, DVB) Project elkészült a DVB-H (handheld, kézi) szabvány első változatával. A szabvány és a rá épülő rendszerek elterjedésével a digitális műsorszórás és a mobil adatátviteli hálózatok közötti különbségek egyre halványabbakká válhatnak, a DVB-H alapú hordozható eszközök földi sugárzással és UMTS hálózatokon keresztül egyaránt képesek lesznek adatok fogadására. Az új műsorszórási megoldásokkal újszerű multimédiás szolgáltatásokat lehet majd nyújtani, például egy focimeccsen a gólokat azonnal vissza lehet játszani, a képet a szurkolók a saját mobiltelefonjukon láthatják majd, mindeközben nem kell igénybe venniük a mobiltelefonos hálózatot. A Nokia máris elkészítette első hordozható DVB-H terminálját, bár az ilyen készülékek tömeges elterjedésére csak 2–3 év múlva lehet számítani.

Frissített SELinux

Új nyilvános változat jelent meg az NSA segítségével fejlesztett Security Enhanced Linuxból. Az új kiadás 2.6.3-as és 2.4.24-es rendszermagokra épül, és számos biztonsági jellegű módosításon ment keresztül. Ugyancsak megújult az NSA honlapja, amelyet teljes egészében átterveztek, s remekül áttekinthető – ámbár vadonatúj fekete-vörös színvilágával egy kicsit fenyegetőnek hat.

➲ http://www.nsa.gov/selinux/

XFree86 4.4

Pontosan egy évvel a 4.3-as változat megjelenése után elkészült az XFree86 4.4-es kiadása. Az ablakozó rendszert nem kevesebb, mint

17 operációs rendszer alá írták meg. Az új változatban számos népszerű VGA-vezérlő illesztőprogramját – elsősorban megbízhatóbb működésre törekedve – továbbfejlesztették, újdonság az IPv6-támogatás, illetve az egérkezelés, a betűtípusok kezelése és a nemzetközi karakterek támogatása terén is számos hibajavítást készítettek.

➲ http://www.xfree86.org

Az Intel is 64 bitre vált

Még ebben az évben az Intel is piacra dobja 64 bites kiterjesztésekkel rendelkező asztali és kisebb kiszolgálókba szánt Xeon processzorait. Az Intel lapkái a programok felől tekintve nagyjából az AMD processzoraival lesznek egyenértékűek, vagyis az alkalmazások a két cég termékein módosítások nélkül is ugyanúgy képesek lesznek futni. Az első ilyen Intel lapka a Nocona kódnevű Xeon DP processzor lesz, amelyet az év második negyedében adnak ki, ezt később, az év közepe táján az új Xeon MP és Pentium 4 processzor követi. Addigra valószínűleg több operációs rendszer – a SuSE, a MontaVista és a Red Hat Linux, Windows – átalakított változata is megjelenik, valamint a szükséges fordítóprogramok is elérhetőkké válnak.

Kicsik, nagyok, gyorsak

Savvio név alatt a Seagate megkezdte 2,5"-os, kiszolgálókba szánt merevlemezeinek a forgalmazását.

A 36 és 73 GB kapacitású, kisméretű meghajtókat elsősorban állványra szerelt kiszolgálókba szánják, amelynél a méret és az alacsony fogyasztás egyaránt fontos szempont. A teljesítmény tekintetében viszont semmiről nem kell lemondanunk, a tízezres fordulatszámon pörgő lemezeket tartalmazó meghajtók nagyobb méretű társaikéhoz hasonló átviteli sebességekre és azoknál jobb elérési időkre képesek. A Savvio meghajtókat többféle SCSI csatolóval lehet majd megvásárolni, a különféle változatok ez év során folyamatosan jelennek meg. Hasonló termékekkel valószínűleg a többi nagy gyártó – Hitachi, Fujitsu, Maxtor – is hamarosan jelentkezik.

Ugyancsak kiszolgálókba készült a Hitachi gigászi meghajtója, a szintén tízezres fordulaton pörgő lemezeket tartalmazó, 3,5"-os Ultrastar 10K300. A Hitachi egysége jelenleg a világ legnagyobb kapacitású kiszolgáló merevlemeze, mint neve is utal rá, kereken 300 GB tárolására alkalmas.

➲ http://www.seagate.com
➲ http://www.hgst.com

Testépítők előnyben

Sokan és sokszor elmondták már, hogy a legújabb noteszgépük „most már aztán igazán” képes az asztali gépek helyettesítésére, de valójában a hordozható gépek soha nem tudtak olyan lehetőségeket biztosítani, mint asztali társaik. A Dell most ugyanezzel a jelszóval nem kevesebbre vállalkozott, mint hogy 9100-as jelzéssel egy minden igényt kielégítő multimédiás, valamint XPS névvel egy kifejezetten játékosoknak szánt, notesznek kevésbé nevezhető divatgépet dob piacra.

A Dell Inspiron 9100-ba 3,2 GHz órajelű Pentium 4 processzor, 400 MHz-es kétcsatornás memória és ATI Mobility Radeon 9700 VGA-kártya kerül, képét 15,4"-os szélesvásznú kijelzőn szemlélhetjük. Multimédiás jellegét a beépített mélynyomó és a Firewire csatoló hivatott biztosítani.

Az XPS akár 3,4 GHz-es processzorával ennél is nagyobb teljesítményt ígér, további érdekessége, hogy fedőlapja négyféle kivitelben kapható: szürke színnel, valamint Cipher, Plasma és Skullz fantázianevű grafikákkal díszítve.

A 4,5 kg súlyú számítógéphez a vásárlók egy tetszetős hátizsákot is kapnak, így bármikor bevetésre készen várhatják a legközelebbi LAN-partyról szóló híreket.

➲ http://www.dell.com

Logikai gyorsítás

A Tarari legújabb terméke – egy PCI-foglalatba illeszkedő kártya formájában megvásárolható, tartalom-

feldolgozó processzor – a programbeli megoldásokhoz képest pontosan negyvenszeres gyorsulást ígér a szabályos kifejezések feldolgozásában. A szabályos kifejezések nemcsak parancsfájlok készítésekor jutnak fontos szerephez, de például tartalomszűrésnél, a levélszemét válogatásánál, a behatolások érzékelésénél, a hálózati forgalom szabályozásában és a házirendek betartatásánál is széles körben alkalmazzák őket.

A bonyolult mintaegyeztető utasítások segítségével számos összehasonlító művelet írható le, ám a feldolgozásuk jelentős számítási teljesítményt igényel – erre kínál gyógyírt a Tarari. Kártyája megfelel a szabályos kifejezésekre vonatkozó Posix 1003.2 szabványnak, a bemenetet többféle írásmóddal képes fogadni, a feldolgozást pedig párhuzamosítva végzi el. A cég hivatalos támogatást Red Hat Linux és Windows Server alá nyújt a kártyához.

➲ http://www.tarari.com/regex

Most már elég pici?

A mobiltelefonok tervezői számára egyre súlyosabb gondot jelent, hogy hiába bővítik újabb és újabb szolgáltatásokkal – fényképezés, mozgókép rögzítése stb. – a készülékeket, az adatok tárolását nem tudják elég hatékonyan megoldani. A meglévő memóriakártyák ugyan kicsik, de nem eléggé – ezért fejlesztette ki a SanDisk a T-Flash kártyákat. A mindössze 15×11×1 mm méretű, a mobiltelefonok akkumulátora alá helyezhető, fizikai hatásoknak ellenálló tárolóegységek először 16 és 128 MB közötti kapacitásokkal jelennek meg, csatolófelületük és biztonsági szolgáltatásaik tekintetében a mostani SD-kártyákkal egyenértékűek. A Motorola máris bejelentette, hogy a legújabb telefonjai támogatni fogják a T-Flash kártyák használatát.

➲ http://www.sandisk.com

Szárnyaló tűzmadár

Több éves fejlesztőmunka eredményeként elkészült a Firebird relációs adatbázismotor 1.5-ös változata.

A Borland 2000-ben tette közzé InterBase adatbázis-kezelőjének forráskódját, bízva abban, hogy a közösségi fejlesztők hozzájárulnak terméke sikeréhez. A független fejlesztők azonban nem fogadták el a Borland feltételeit, és tőle függetlenül folytatták munkájukat, igaz, megőrizve az együttműködés lehetőségét az InterBase újabb változataival. Az 1.5-ös változat megjelenése fontos mérföldkő a tervezet életében, de a munka nem áll meg, a következő lépés a tucatnyi operációs rendszer alá elérhető adatbázis-kezelő 2-es változatának elkészítése.

Ide tartozik, hogy május 16–18. között Németországban, vagyis a magyar fejlesztők számára is könnyen elérhető helyszínen rendezik meg a második nemzetközi Firebird-konferenciát.

➲ http://www.firebirdsql.org

Mini iPod

Az Apple megkezdte iPod minikészülékeinek a szállítását. Az elődjéhez képest feleakkora súlyú és méretű MP3-lejátszó készülék 4 GB memóriával rendelkezik, s ezáltal körülbelül 1000 zeneszám tárolására alkalmas. A nagynak

csak a mini mellett mondható, körülbelül 150 gramm súlyú eredeti iPod kapacitása ugyanakkor megnőtt, a csúcsmodellt vásárlók immár 40 GB memóriával gazdálkodhatnak. Az új mini ötféle színben kapható, utcai ára 249 dollár, vagyis nagyjából 54 ezer forint.
➲ http://www.apple.com/ipodmini

Linuxos zenei konferencia

Április 29. és május 2. között kerül megrendezésre a németországi Karlsruhe városában a második Nemzetközi Linux Audio Konferencia. A rendezvényre a világ számos országából érkeznek előadók, az érintett gyártók kiállításra kerülő termékei pedig a zeneszerzésről egészen a gyártásig minden területre megoldásokat nyújtanak majd – ezek tudásáról koncertek, bemutatók során bizonyosodhatnak meg az érdeklődők. A konferencián való részvételhez előzetes jelentkezésre nincs szükség.
➲ http://www.zkm.de/lad

Shavlik: központi folttelepítés

A Shavlik Technologies HFNetChkPro terméke jó ideje elismert megoldás a Windowsok frissítéseinek önműködő, központi telepítésére, ám most a cég egy linuxos megoldást is meg kíván jelentetni. A Shavlik nemrég vásárolta fel a Gibraltar Software-t, s az így megszerzett eszközök felhasználásával egy ügynökalapú rendszert fognak kidolgozni. Ennek segítségével a helyi hálózatok Linux alapú gépei rendszeresen lekérdezhetik a kiszolgálótól a rendelkezésre álló foltok listáját, a kiszolgáló pedig ellenőrizheti, hogy az ügyfeleken szükség van-e valamilyen javítás telepítésére.
➲ http://www.shavlik.com

Nem tetszett, kihúzták

Egy holland bíró ítélete alapján a Lindows márkanév túlságosan hasonlít a Windowsra, ezért a Benelux államokban megtiltották a használatát. Az ítélet azért is érdekes, mert a tengerentúlon mindezt pontosan fordítva gondolták, ott úgy vélik, hogy a windows szó – mint „ablakok” – egyszerű köznév, amely a Microsoft termékének megjelenése előtt is létezett és ismert volt. A maga szempontjából mindkét bírónak igaza van, hiszen az angol anyanyelvűek számára az ablak hétköznapi szó, míg a más nyelvet beszélőknek a windows csak egy angol karaktersorozat, amely az ő nyelvükön – eredetileg – nem jelent semmit, ezért akár védjegyként is kezelhető.

A tiltó határozatra válaszul a Lindows.com a Hollandiában, Luxemburgban, Belgiumban és Svédországban élők számára új nevet vezetett be: LinDash (nagyjából ***Linvonás***), vagyis Lin---s. A ➲ http://www.lin---s.com címen látható akasztófa-játékos rajz nagyjából azzal egyenértékű, mintha a Lindows.com képviselője nyerítve röhögne a bíró szemébe, hiszen tökéletesen kifejezi az amúgy nyilván jogszerű bírói döntés értelmetlenségét. Hiába ugyanis a tiltás, ha a Lindows márkanevet már mindenki ismeri, „úgyis mindenki tudja, miről van szó”, és az eredeti lindows.com című weboldal elérését sem akadályozza semmi az említett országok lakói számára. A termék értékesítését tehát továbbra sem gátolja semmi; a legfontosabb piacon, az Egyesült Államokban pedig még a névhasználat kapcsán sem tudják akadályozni a Lindows menetelését.
➲ http://www.lin---s.com

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

Ők mondták

Határozottan ellenzem, hogy olyan szavazórendszert vásároljunk vagy használjunk, amelynek a forráskódja és a tervezése nem nyilvános. Bizonyára vannak olyan vállaltok, amelyek szívesen választják ezt a modellt, különösen, ha a szerződés bizonyos hosszú távú kötelezettségeket is tartalmaz. Ez nem egy Britney Spears-súlyú ügy – a szavazati rendszer épsége a kormányzat alapvető alkotóeleme.
Phil Windley, ➲ http://www.windley.com/2003/11/03.html#a893

Az elektronikus kormányzati alkalmazások nem lehetnek drágák, lehetővé kell tenniük a gyors fejlesztést, felhasználóbarát környezetet kell teremteniük a választópolgárok számára és fejlett adatvédelmi szolgáltatásokkal kell rendelkeznie. A Linux minden olyasmit birtokol, amire az elektronikus kormányzati kezdeményezéseknek szüksége van.
Tom Adelstein, ➲ http://www.linuxjournal.com/article/7076

Azonnal megkezdjük a Java Desktop System terjesztését és 2004 folyamán 1–1,5 millió asztali számítógépet fogunk meghódítani… Ezzel egy csapásra a miénk lesz a világ első számú linuxos asztali rendszere.
Scott McNealy, Business Standard
➲ http://www.business-standard.com

A Linux-kiszolgálókat üzemeltetők hat egymást követő negyedéven keresztül folyamatos bevételnövekedést könyvelhettek el, ami egyértelműen bizonyítja, hogy nem hirtelen fellángolásról van szó, és megfelelnek a nagyteljesítményű informatikai megoldások (HPC) és kereskedelmi célú fejlesztések valós számítástechnikai igényeinek.
Jean Bozman, IDC
➲ http://www.askwebhosting.com

A jövőben az alapvető üzleti alkalmazások széles palettájú linuxos megvalósítása talán kisebb nyilvánosságot fog kapni, de folyamatosan jelen lesz a világ vezető pénzügyi szervezeteinek mindegyikében… Ehhez az átrendeződéshez elengedhetetlenül szükség lesz a nagy számítógépgyártók folyamatos támogatására. Úgy gondoljuk, hogy az elkövetkezendő 18–24 hónapon belül a Linux alapú programcsomagok telepítése ugyanolyan természetessé válik, mint egy Windows vagy Sun Solaris rendszer telepítése.
Damon Kovelsky, IDC
➲ http://www.internetnews.com/ent-news/article.php/1559661

Nincs mentség a bináris modulok számára. Néhányuk talán elfogadott (abból adódóan, hogy önálló fejlesztés) és megengedett, de még az ilyenek is rengeteg bosszúságot okoznak, és számtalan hibalehetőséget rejtenek.
Időről-időre panaszok érkeznek hozzám, amiért nem is próbálom segíteni a bináris modulokat. Ez van. Ez kölcsönösen működik: ha te nem segítesz nekem, én sem segítek neked. A csak bináris modulok egyáltalán nem segítik a Linuxot, és így van ez fordítva is. Ilyenformán semmi nem ösztönöz minket arra, hogy annál nagyobb mértékben segítsük az ilyen modulok terjedését, mint amilyen a jelenlegi.
Linus Torvalds a linux-kernel levelezési listán

Linux Journal 2004. március, 119. szám

Linux-index

1. Ennyi ezerre tehető azoknak az amerikai helyhatóságoknak a száma, amelyek készek nyílt forrású alkalmazások bevezetésére: 88
2. Az Egyesült Államok haditengerészete legkevesebb ennyi dollárt takarított meg egyetlen nyílt forrású együttműködési fejlesztéssel: 300 000
3. Ennyi ezer Linux-ügyfél telepítését tervezi a Sherwin-Williams Co.: 10
4. Várhatóan ennyi ezer Linux alapú Sun Java Desktop rendszert fognak telepíteni egy terítési folyamat keretében az Egyesült Királyság területén: 800
5. Legkevesebb ennyi millió Linux alapú Sun Java Desktop rendszer telepítése várható évente Kínában: 1
6. A Sun ennyi millió Linux alapú Sun Java Desktop rendszer telepítését tűzte ki célul Kínában: 500
7. A helyhatósági informatikai költségkeretek ennyi százalékát „emésztik fel a hatékonyságnövelő programcsomagok és eszközök”: 60–70
8. Ennyi a Linux alapú, nyílt forrású, hatékonyságnövelő programcsomagok és eszközök tervezett költsége a helyhatósági költségkeretek terhére: 0
9. Az ügyvezető igazgatók és informatikai vezetők ennyi százaléka számolhat be Linux-kiszolgáló alkalmazásáról Ausztráliában és Új-Zélandon: 32,4
10. Évről évre ennyi százalékkal növekszik a Linux-kiszolgálók eladási aránya: 51,4
11. Ennyi millió dolláros bevételt hoztak a Linux-kiszolgálók az elmúlt negyedévben (2003. november előtt): 743
12. Ennyi lesz a Linux részesedése a kiszolgálói piacon legkésőbb 2007-re: 45%

Források
1., 2., 7., 8.:Open Source Software Institute
3.: LinuxInsider
4.: The Register
5.: CNET Asia
6.: vnunet, a Sun vállalatvezetőitől
9.: CNET
10–11.: Ask Web Hosting, IDC
12.: The Inquirer, Meta Group

Linux Journal 2004, 119. szám

Rendszermag-fejlesztési hírek

Linus Torvalds már több 2.6-os próbakiadás során utalt arra, hogy a fát hamarosan *Andrew Morton*-nak adja át. Ez a 2.6.0-test11 kiadással hivatalossá vált, Andrew látja el a 2.6-os fa karbantartási feladatait. Ez különösen érdekes abból a szempontból, hogy a 2.6.0-s változat kiadása jelentős esemény volt, és Linus helyett Andrew került az érdeklődés középpontjába. A lépés szépen egybevág azzal, amit Linus szokott válaszolni a nélkülözhetetlenségét hangoztató kijelentésekre. Linus mindig is ragaszkodott ahhoz, hogy ő „csak egy a fejlesztők közül", és egyáltalán nem az a központi személyiség, amivé az emberek szemében az elmúlt időkben vált. Ezzel a félrevonulással Linus valószínűleg a saját nyilvános szerepét próbálja csökkenteni.

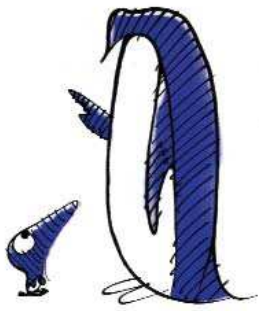
Elméletben a megbízható rendszermagváltozatok célja a valódi megbízhatóság fokozatos elérése. A gyakorlatban a folyamatban lévő fejlesztési változat szolgáltatásait valamennyi próba után gyakorta visszamenőlegesen beveszik a megbízható változatba. Bizonyos esetekben a fejlesztőmunka közvetlenül a megbízható változatban zajlik, ha a módosítások tisztán elkülöníthetőek. Azonban általában véve elmondható, hogy a tiszta megbízhatóság mint végső cél egyre erősebbé válik. Most, hogy a 2.6-os fa karbantartását Andrew Morton vette át, *Marcelo Tossatti* úgy döntött, hogy lezárja az új szolgáltatások bevételét a 2.4-es fába, és eltekintve néhány esetleges kivételtől, mint az XFS-támogatás, a 2.4-es rendszermag hamarosan csak hibajavításokat és biztonsági javításokat fogad be.

Egyesek alapvető változásnak tarthatják azt, amin *Pontus Fuchs* munkálkodik jelenleg: bizonyos Microsoft Windows-vezérlőprogramokat próbál meg működésre bírni a Linux rendszermagjában. Figyelembe véve, hogy ezek a vezérlőprogramok teljesen más operációs rendszerhez készültek, meglepően sikeresen halad a fejlesztés. Egyes vezeték nélküli LAN-kártyák nem rendelkeztek sem leírással, sem linuxos vezérlőprogrammal, akár szabad program, akár csak bináris modul formájában. Pontus Fuchs mostanára elérte, hogy sikeresen tudja használni saját Broadcom 4301-es kártyáját, és *Pavel Machek* ugyancsak működőképessé tette a Broadcom 94306-ost. *Carl-Daniel Hailfinger*, *Manfred Spraul* és *Andrew de Quincey* az nVidia nForce MCP ethernet-illesztőkártyához tartozó `nvnet` vezérlőprogram alapján, visszafelé történő fejlesztéssel írta meg a saját GPLed vezérlőprogramját. A tiszta fejlesztési környezet megőrzése érdekében, amely kizárja, hogy az nVidia vezérlőprogramjából bármilyen részt átmásoljanak a saját GPLed programjukba, Carl-Daniel és Andrew végezte a visszafelé történő fejlesztést, és elkészítették az alkatrész leírását, Manfred pedig ez alapján megírta az új vezérlőprogramot. Így biztosították, hogy Manfred nem ismerte közvetlenül az eredeti vezérlőprogram belsejét. A visszafelé történő fejlesztés hagyományos számítástudományi módszer az együttműködés elérésére, habár sok országban erősen próbálkoznak azzal, hogy törvény tiltsa a visszafelé történő fejlesztés bizonyos formáit. A DMCA (Digital Millennium Copyright Act) is efféle rendelet, amely korlátozni próbálja a visszafelé történő fejlesztést, de hogy valójában milyen hatása lesz, az a bíróságokon fog eldőlni.

Megváltozott néhány jelkezelési tulajdonság a 2.4-es és a 2.6-os rendszermag között. Vannak olyan jelek (az úgynevezett szálszinkron-jelek), amelyeket a programszálak nem tudnak megakadályozni. A 2.4-es rendszermagban sikertelen volt az a szál, amely egy ilyen jelet gátolni próbált, a jel pedig mindenképpen célba ért. Ez nem feltétlenül kívánatos viselkedés, mivel igazából nincsen elfogadható indok arra, hogy egy szál megakadályozza ezeket a meggátolhatatlan jeleket. Ezért a 2.6-os rendszermagban megszakad az a programszál, amely szálszinkron-jelet próbál elnyomni. Így a programozó azonnal látja, hogy valami olysmi történt, aminek nem lett volna szabad. Linus szerint az új módszer a legjobb módszer, de sok múltbeli „legjobb módszer" még jobb módszerekhez vezetett. Valószínűleg ebben az esetben is ez fog történni.

Zack Brown

Linux Journal 2004. március, 119. szám



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

A hónap szakmai tanácsai

A legjobb program biztonsági mentésre?

Amikor egy újságárusnál belelapoztam Linux Journalba, el sem tudtam hinni, hogy számos addigi gondomra egy cikkben megoldást találok. Azonnal megvettem azt a számot és amint hazaértem, előfizettem a lapra az interneten keresztül. Miután feldolgoztam a Nagiosszal nyert ismereteket, most a cégünk jelenlegi DLT-mentésének a kiváltására keresek megoldást. Jelenleg egy Windows alatt futó Veritas Backup Exec 8.6-ot használunk és súlyos pénzeket fizetünk érte. A kérdésem, hogy létezik-e olyan valamirevaló linuxos megoldás, ami a szalagos backup eszközök széles választékát támogatja? Jelenleg négy DLT 15/30 GB meghajtót használok az irodában, de az otthoni hálózatomban szükségem lenne egy hétkazettás DLT kazettaváltó támogatására is. Valami olyat szeretnék, aminek használatát rövid idő alatt el lehet sajátítani. Mivel egy átlagos felhasználónak két egérkattintásba kerül elveszítenie egy fájlt, legalább olyan könnyen szeretném azt visszaállítani.
Eric Patat, epatat@charter.net

A BRU (Backup and Recovery Utility) meglehetősen ismert és elismert kereskedelmi program, de nem túl drága. További tájékoztatást a ➲ http://www.tolisgroup.com címen lehet találni. A BURT (BackUp and Recovery Tool) a Wisconsini Egyetemen (➲ http://www.wisc.edu/~jmelski/brut), az Amanda pedig a Marylandi Egyetemen készült.
Jim Dennis, jimd@starshine.org

Két népszerű programot tudok ajánlani, az egyik az Amanda (➲ http://amanda.org), ez szabadon felhasználható, a másik az Arkeia (➲ http://www.arkeia.com), ez kereskedelmi termék.
Don Marti, dmarti@ssc.com

A backup program nem szükségszerűen a legjobb ok arra, hogy Linuxra váltsunk. Nem azért, mert Linux alatt nincs ilyen, hanem azért, mert gyakran ugyanazt a terméket találjuk itt is. A legtöbb nagy kereskedelmi programgyártócég támogatja a Linuxot. Léteznek középkategóriájú megoldások is, amelyek költséghatékonyabbak, de még van grafikus kezelőfelülete és varázslója. Ha a nyílt forrású megoldásokat részesíted előnyben, széles választékot találhatsz ezen a területen is, de egyszerűen a jó öreg `tar` és `gzip` párost is használhatod, vagy valami erőteljesebbet, például a `cpio`-t. Szükséged lesz az `mt` mágnesszalag eszközcsomagra és a megfelelő meghajtóprogramra. Ha szalagtárat (tape library) használsz, keresni kell egy olyan segédprogramot, amely vezérli az eszközt, szóval, mielőtt vásárolsz valamit, érdemes utánajárni a dolgoknak.
Chad Robinson, crobinson@rfgonline.com

Úgy találtuk, hogy az átlagtól eltérő backup-szükségleteinkre messze az a legjobb megoldás, ha a mentésre, a mentés ellenőrzésére, érvényesítésére és a visszaállításra saját parancsfájlokat (jobbára Pythonban készülteket) használunk. Adat- és lemezmentést is végzünk szalagos egységre, CD alapú eszközre és másik merevlemezekre egyaránt. Néhány referencia a ➲ http://www.linux-backup.net címen érhető el, ahol különféle adatokat lehet találni a Linux alatti adatmentésről, ezenkívül nézz bele a „Unix Backup and Recovery" könyvbe, amit régebben a Linux Journal is bemutatott. Bár a könyv egy kicsit öreg, talán mégis érdemes elolvasni. A Linux Journal-bemutató a ➲ http://www.linuxjournal.com/article/3839 címen olvasható. Az eszközökkel kapcsolatban a ➲ http://www.linuxtapecert.org webhelyen nézz körül.
Felipe Barousse Boué, fbarousse@piensa.com

Rejtsd el azt a jelszót!

Valószínűleg nem én lehetek az egyetlen, akivel gyakran előfordul az alábbi kellemetlen helyzet. Bármilyen okból kifolyólag is, de vannak olyan esetek, amikor véletlenül jelszót írok a parancssorba, azt gondolván, hogy az egy program, ami például az `ssh` vagy az `smbclient stdin` bemenetére megy. A bash héjprogramot használom, ami azt jelenti, hogy óvatlanságom eredménye bekerül a naplófájlba. Ez általában nem nagy gond, de néha megosztott felhasználói azonosítóval dolgozom egy rendszeren. Mondanom sem kell, hogy aki még hozzáfér ahhoz az azonosítóhoz, megtalálhatja a jelszavamat a naplófájlban. Létezik valami egyszerű módja, hogy megmodjam a bashnak, hogy mellőzze az előző vagy bizonyos bejegyzéseket a naplófájlban? Szeretném, ha nem kézzel kellene szerkesztenem a naplófájlokat, bár egyelőre ez látszik számomra az egyetlen lehetséges módszernek, hogy megtisztítsam őket.
Chris DeRose, cderose@deroseandslopey.com

Először is, ha észreveszed a tévedést még az ENTER lenyomása előtt, egyszerűen használd a CTRL-U-t. Ez mindent töröl a pillanatnyi parancssorból. Ez működik a parancshéj promptjában (a legtöbb Bourne-héjnál), a bejelentkező promptban, még `vi`-ben is (mialatt továbbra is beszúró módban vagyunk). Ha már megnyomtad az ENTER-t, a leggyorsabb és legkönnyebb módszer a már lemezen lévő naplófájl újraolvasása. Mivel a naplófájl normál esetben csak kijelentkezéskor kerül kiírásra, ez felül fogja írni a memóriában lévő naplót. Írd be a `cat>~/.bash_history` parancsot. Ez természetesen az eddigi munkamenet minden bejegyzését ki fogja törölni, és azt az állapotot kapod vissza, ami a bejelentkezéskor állt fenn.
Jim Dennis, jimd@starshine.org

A `history` parancs lehetőségeinek a megtekintésére használd a `man history`-t:

```
history [n]
history -c
history -d eltolás
```

A `history` parancs kapcsoló nélkül a parancsnaplólistát jeleníti meg, sorszámozva a sorokat. Az n szám megadásával az utolsó n sort lehet kiíratni. A `-c` vagy `-d` kapcsoló, amennyiben működik, a következőt teszi: a `-c` az összes naplóbejegyzést, a `-d` eltolás pedig az eltoláspozícióban lévő bejegyzést törli.
Felipe Barousse Boué, fbarousse@piensa.com

Slackware Serial ATA rendszeren?

Slackware 9.1-est szeretnék telepíteni egy új rendszeren, amely serial ATA-meghajtót használ. A rendszerben egy Intel alaplap van két IDE kapuval és két serial ATA kapuval. Összesen hat eszközt lehet a géphez kapcsolni. A két IDE kapura két CD-ROM-meghajtó és egy zipmeghajtó csatlakozik. Egy SATA kapun található a rendszer egyetlen merevlemeze, a másik SATA kaput nem használom. A rendszert a Slackware indító-CD-jéről indítom, az indítás rendben lezajlik egészen addig a pontig, amíg a meghajtókat deríti fel. A rendszer az összes kaput látja, és **hde**-ként látja a SATA-meghajtót az IDE2 kapun. A lemezrészekre való felosztás után a **hde4** lemezrészt kell a Slackware-nek felismernie. A rendszer tudja, hogy a meghajtó ott van, de az alábbi üzenettel áll meg:

```
hde4: loading IDE drivers
```

Ennél a pontnál nem sikerült továbbjutnom. Hogyan tudnám Linux alatt használni a SATA-meghajtót?
Ren Colantoni, colanton@lacitycollege.edu

Eszköztől és terjesztéstől függő trükköket kell bevetni a rendszer használatához. Hogy miket, annak felderítéséhez jó kiindulópont a ➲ http://deploylinux.typepad.com/main/2003/07/linux_sata_supp.html oldal. Itt lehet a vezérlő, a meghajtó és a Linux együttműködési gondjainak a megoldására tippeket találni.
Mario Bittencourt, mneto@argo.com.br

A legjobb eszköz egy WordPerfect-szakértőnek?

Több mint két évtizede használok valamilyen formában Linuxot. Red Hatet használva a „The Economy and Material Culture of Russia 1600-1725" című, legutolsó könyvemről kefelenyomatot készítettem. Ez egy nagy formátumú, 668 oldalas, 104 grafikonnal tűzdelt kiadvány, ami a Stata segítségével készült 108 000 filePro16 adatbázis-bejegyzésből. Ezt a kefelenyomatot a kivételesen ügyes WordPerfect szövegszerkesztővel készítettem. Most az egyetemen frissíteni kell a számítógépemet, amin Red Hat Linux 9 lesz. Tudom, hogy a Corel nem foglalkozik már a WordPerfect karbantartásával, ezért az nem fog futni a Red Hat Linux 9-en. Melyik a leginkább használható szövegszerkesztő csomag ehhez a projekthez?
Richard Hellie, hell@midway.uchicago.edu

Számos szövegszerkesztő található, a választás pedig a megjelenési, kiadási igényektől függ. A vizsgálódást a mindig jelenlevő Emaccsal, valamint a LaTeX és SGML dokumentumleíró nyelvekkel kezdheted. Egy csomó ember úgy találja, hogy ezeket túl bonyolult a saját igényekhez igazítani, de mindig megéri megnézni, s ha tudod, hogyan kell használni őket, rendkívül hatékony dokumentumfeldolgozó eszköznek bizonyulnak. Ha inkább a WYSIWYG (azt kapod, amit látsz) szövegszerkesztőt szeretnéd használni, akkor telepítheted az OpenOffice.org-ot vagy a Kwordöt, mindkettő nyílt forrású program. Ha a Microsoft Office-felhasználókkal való nagyobbfokú együttműködésre van szükséged, kipróbálhatod akár a Sun StarOffice termékét, ami tulajdonképpen az OpenOffice.org többek között betűtípusokkal és kereskedelmi támogatással bővített változata; akár az IBM Lotus SmartSuite-ját, ami szintén kereskedelmi termék. Ez csak néhány az elérhető lehetőségek közül, és még nem is tartalmazzák az asztali kiadványszerkesztő termékeket. Nézz körül, meg fogsz lepődni az elérhető lehetőségek számától.
Chad Robinson, crobinson@rfgonline.com

Mint más nem támogatott zárt forrású programot, a WordPerfectet is lehet futtatni a könyvtárak régi változatainak telepítésével (➲ http://www.linuxmafia.com/wpfaq). Ha importálni szeretnéd a Microsoft Word-dokumentumokat, telepíteni kell egy javítást is, amely elérhető a ➲ http://www.linuxjournal.com/article/5655 címen.
Don Marti, dmarti@ssc.com

Nem működik a leállítás

Két operációs rendszer fut a számítógépemen, egy Microsoft Windows XP és egy SuSE 8.2. Amikor az XP-t leállítom, a PC kikapcsol, de amikor a Linuxot állítom le, a PC újraindul.
Bouve, andre.bouve@pandora.be

A jelenséget valószínűleg az SMP és az APM között lévő hiba okozza. A két szabvány nem működik együtt, ami a több processzor közötti elkerülhetetlen versenyhelyzetben nyilvánul meg. Linuxon az SMP rendszermagban az APM szolgáltatás ki van kapcsolva, még akkor is, ha éppen egyprocesszoros gépen fut. Megoldásképpen megpróbálhatod egyprocesszoros rendszermagra cserélni, vagy olyan kapcsolókkal fordítani a rendszermagot, amely erőlteti az APM-kikapcsolás működését.
Jim Dennis, jimd@starshine.org

Próbáld meg rendszerindításkor az `apm=power-off` értéket átadni a rendszermagnak.
Usman S. Ansari, uansari@yahoo.com

Linux Journal 2004. március, 119. szám

Új termékek

LC2430 Debian noteszgép

A LinuxCertified Inc. bejelentette első, Debian-tanúsítvánnyal ellátott noteszgépének, az LC2430-nak a megjelenését. A Debian-modell a legújabb az LC2000 sorozatból és előre telepített Debian GNU/Linuxszal szállítják. A Unix munkaállomás kiváltására szolgáló LC2430-as SXGA+ képernyővel, legfeljebb 3,06 GHz-es Pentium 4 HT (hyperthreading) processzorral, legfeljebb 2 GB RAM-mal és 80 GB merevlemezzel kerül forgalomba. Jellemzője még az Ati Mobility Radeon 9000 grafikus lapka 64 MB memóriával, OpenGL gyorsítással, a kombinált DVD-olvasó és CD-RW-író meghajtó, valamint a 10/100 Mbit/s-os hálózati csatoló. A noteszgépen végzett Debian GNU/Linux-teszt elsősorban a telepítést, a beállítást és az üzemeltetést vizsgálta, erről részletek a LinuxCertified weboldalon találhatók.
➲ http://www.linuxcertified.com

JMP 5.1

A JMP 5.1-es változata egy Linux-változatot is tartalmaz az asztali statisztikai elemzőprogramból. A JMP 5.1-essel a statisztikai elemzések dinamikusan összekapcsolhatók az adatok megjelenítését végző grafikonokkal. A beépített modellezési lehetőség abban segít a felhasználóknak, hogy megtalálják a hiányosságok gyökerét olyan esetekben, amikor több változó nemlineáris kapcsolatban áll egymással, ha nem ismerhető fel a modell, vagy amikor az alaptényezők mérése kimaradt az adatok közül. A JMP tartalmazza a Six Sigma hagyományos és testreszabott DOE (Design Of Experiment) eszközeit. Az 5.1-es új statisztikai alapjai lehetővé teszik a szorosan egymáshoz kapcsolódó, valamint a nagy mennyiségű adatok analizálását is.
A JMP 5.1 támogatja a Red Hat, a Red Hat Advanced Server, a SuSE, a Mandrake és a UnitedLinux-változatokat.
➲ http://www.jmp.com

Centrus földrajzi csomag

A Group 1 Software bejelentette Centrus nevű kereskedelmi földrajzi csomagját. A Centrus termékek egyesítik a beágyazható technológiát és a kívülálló adatforrások széles választékát olyan feladatok megoldása érdekében, amelyeknél az elhelyezkedés alapvető fontosságú. A Centrus GeoStan rendszer, amely a címadatokat javítja és térbeli adatokkal egészíti ki, most olyan kereskedőiszakasz- (vendor segment) azonosítóval bővíti a kimenő adatokat, ami a vezető kereskedők adataihoz kapcsolódik. Tömbtoldalékok (block suffixes) is megjelennek a kimeneten az US Census Bureau TIGER 2002 adatait felhasználva. A Centrus Address-Broker felépítése korszerűsödött, így ki tudja használni egy több processzoros linuxos gép előnyeit; ráadásul az összes Centrus termék fut Red Hat Linuxon.
➲ http://www.g1.com

AccuPoll

Az AccuPoll megjelentette új, többnyelvű elektronikus szavazórendszerét, ami az érintőképernyős bemenet átláthatóságát a szavazó által ellenőrzött, nyomtatott papíros jelentéssel kombinálja. A szavazóállomás végigkalauzolja a szavazót a szavazás folyamatán, a szavazatok pedig egy képernyőn történő jóváhagyással és egy, a szavazóállomás által papírra nyomtatott szavazási igazolással kerülnek megerősítésre. Amint megszületett a szavazat, az AccuPoll rendszer egy független, a szavazó által ellenőrzött audit bejegyzést végez, ami nyomtatott és elektronikus formában egyszerre több helyen is rögzítésre kerül. Az AccuPoll rendszer megfelel a 2002-es Help America Vote törvény követelményeinek, csakúgy, mint a mozgásában vagy egyéb módon korlátozott szavazók igényeinek. Az AccuPoll nem egyedi eszközökön fut és nyílt forrású program.
➲ http://www.accupoll.com

VxWorks együttműködési réteg

A LynuxWorks egy VxWorks együttműködési réteg (Compatibility Layer) csomagot kínál, hogy segítse a VxWorks-kódok átültetését (portolását) az ő LynxOS valós idejű operációs rendszerére (RTOS). A csomag használatával az eredetileg VxWorks folyamatos memóriamodelljére (flat-memory model) írt alkalmazások a LynxOS többszálú, Posix-megfelelő valós idejű operációs rendszerén használhatók. A VxWorks együttműködési réteg egymástól elválasztott névtereket kezel, ami több virtuális VxWorks-környezet egymás melletti futtatását teszi lehetővé, amikor ez szükséges. A VxWorks átültetőcsomag (porting kit) emellett útmutatást nyújt az olyan típusú kódok felismeréséhez, amelyek különös figyelmet érdemelnek.
➲ http://www.lynuxworks.com

X-treme Alerts Platform

A 724 Solutions – egy következő nemzedékbeli, IP alapú hálózat- és adatszolgáltatást nyújtó szolgáltató cég – Linuxon kínálja X-treme Alerts Platform (XAP) nevű termékét. Az XAP olyan figyelmeztetésre, riasztásra szolgáló felületi eszköz (actionable alerting platform tool), ami a mobilhálózaton dolgozó telefonkezelő számára lehetővé teszi, hogy könnyen azonosítható, engedély-alapú SMS-eket, illetve MMS-eseket küldjön az előfizetők figyelmeztetése végett. Az XAP legfrissebb változata, ami bérelhető vagy a megrendelőhöz telepített formában is elérhető, nagyobb teljesítményt és alacsonyabb költséget (TCO – Total Cost of Ownership) nyújt a mobilhálózat kezelőinek. Az XPA a 724 Solutions X-treme Mobile Suite része, ami következő nemzedékbeli adathálózati megoldást nyújt, lehetővé téve differenciált, személyre szabott, feláras adatszolgáltatásokat.
➲ http://www.724.com

Linux Jounal 2004, 119. szám

Informatikai bizonyítvány egy kicsit másképp

Hogyan lehet egy bizonyítvány jobb, szélesebb körben elterjedt, olcsóbb és még független is?

Az 1998-ban létrejött Linux Professional Institute (LPI) egy olyan, Linux-bizonyítványokat kibocsátó szervezet, amelynek az a célja, hogy elhárítsa az akadályokat a nyílt forráskód terjedése elől és növelje a közösség tudásszintjét. Megalakulása óta az LPI több mint 35 000 vizsgát bonyolított le a világ szinte minden országában, és terjesztők, oktatók, munkaadók, valamint kormányok támogatását nyerte el. Ez a siker azonban csak a kezdet. Az LPI mint szervezet rendelkezik azzal a képességgel, hogy a feje tetejére állítsa az informatikai minősítések világát, éppúgy, ahogy a Linux maga a programok világát forgatta fel.

Manapság mindenki természetesnek veszi, hogy az informatikai bizonyítványok az informatikai cégekhez kötődnek, ami kizárólag ezen a területen ismert jelenség. A gyógyszerészeket például nem úgy képezik ki, hogy csak egy cég gyógyszereit ismerjék. Még az olyan „semleges” programok esetében is, mint a CompTIA, a cégek tartják a kezükben a minősítési folyamatot.

A bizonyítvány megszerzésére törekvő embereknek és leendő munkaadóiknak nincs beleszólásuk a képzés tartalmába. Noha az LPI céljai megvalósulása érdekében elfogadott céges támogatásokat, a cégek mégsem határozhatják meg számára a követendő utat. Az LPI mögött közösségi hajtóerő áll, és nem csupán semleges, hanem független is a szállítóktól. Például a közösségben igény merült fel a felhasználói Linux-bizonyítványra, ezért az LPI megkezdte azt a munkát, ami ezt valóra váltja. Még ennél is fontosabb közös erőfeszítésünk, hogy minden kapálózás és jajveszékelés ellenére az informatikai-bizonyítvány-osztókat afelé az egymás ellenőrzésén alapuló modell feléráncigáljuk, ami más foglalkozások esetében már elvárásnak számít.

Miközben a legtöbb cég fölfelé srófolja bizonyítványt adó vizsgái árát, az LPI tudatosan törekszik a vizsgaköltségek lefaragására.

Az LPI egyik fő törekvése az a kutatási és fejlesztési tevékenység, amelynek célja a számítógépes oktatás költségeinek csökkentése, valamint a vizsgáztatás műszaki hátterének fejlesztésével egy méretezhető, költséghatékony gyakorlati vizsga bevezetése. Növekedésünk a Linux és a nyílt forráskód fejlődésének függvénye, ezen a téren pedig jelentős eredményeket tapasztalhattunk a vállalati és a közszférában egyaránt.

Noha nem lehet kétséges, hogy a Linux az egész világon jelen van, komoly erőfeszítés szükséges ahhoz, hogy a nyílt forráskódhoz kapcsolódó oktatás és a bizonyítványok a számos különféle nyelvű, kultúrájú és kormányzati berendezkedésű országban megfelelőek és hasznosak legyenek. Az LPI a központi irodájából nem képes egymaga megfelelő megoldássá tenni a Linuxot az egész világon. Nem működne a hagyományos kirendeltségekre épülő megoldás sem. Közösségi alapú partnerhálózatunk működik az Egyesült Államokban, Japánban, Brazíliában, Kanadában, Németországban és Ausztriában. Napról napra újabb nemzeti partnerszervezetek kapcsolódnak be; a legutóbbiak között van például Ausztrália, Nigéria, Bulgária és Kína.

Rövid távú célunk a partnerhálózat kiterjesztése, de van egy hosszú távra szóló álmunk is: biztosítani, hogy a minősítő vizsgák elérhetőek és a lehető legjobb minőségűek legyenek, és mindenhol a világon megfelelő megoldást jelentsenek.

Hosszú távú célunk: a vállalatoknak tudniuk kell, hogy amennyiben a Linux mellett döntenek, a szakértelem képzett, minősített szakemberek személyében a rendelkezésükre áll, s ők maguk is teljesítették az LPI programot.

Csüggesztő, ha az ember valamelyik termék újabb változatának a megjelenésével elveszti bizonyítványa érvényességét, és az LPI tisztában van vele, hogy a bizonyítványok megújítása éppolyan fontos az informatikai szakmában, mint bármely más területen.

A különbség az LPI és más informatikai bizonyítványok között abban áll, hogy amennyiben bevezetjük a bizonyítványok megújítását, annak alapja az eltelt évek meghatározott száma lesz, nem pedig egy termék új változatának a kiadása.

Mint maga a Linux, az LPI minősítési programja is a közösségi részvétel erejét bizonyítja. Egyetlen más minősítési program sem egyesít ilyen világméretű, alulról szerveződő hálózatot a vizsgák fejlesztése, a pszichometria és a lebonyolítással foglalkozó szakemberek szaktudása révén. Azt szeretnénk, hogy az LPI a továbbiakban is az a példa lehessen, amit előbb-utóbb minden más informatikai minősítési programnak követnie kell.

Linux Journal, 117. szám

Evan Leibovitch
A Linux Professional Institute enöke.
➲ http://www.lpi.org

Perl-programok teljesítménynövelése

Négy alaplépés, amellyel elkészült programjainkat felturbózhatjuk.

Fejlesztőtársam és jómagam egy olyan adatgyűjtő programon dolgoztunk, amelyik jobbára Perl nyelven íródott. Az alkalmazás mérési adatfájlokat gyűjtött össze könyvtárakból, értelmezte őket, elvégzett néhány statisztikai számítást, majd az eredményeket adatbázisba írta. Ahhoz azonban, hogy a termelési időszak alatt várható jelentős terhelést is elviselje, programunk teljesítményét meg kellett növelnünk.

E cikkből megismerhetjük a teljesítménynövelés négy alaplépését: az azonosítást, a teljesítménymérést, az újraírást (refactoring) és az ellenőrzést. Teljesítménynövelés céljából ezeket a lépéseket alkalmazhatjuk egy meglévő programra. Megállapítjuk, hogy melyik függvény okozhat teljesítménycsökkenést, majd alapszintű teljesítménymérést végzünk rajta, különféle teljesítménynövelő módosításokat hajtunk végre, végül a változtatások eredményét összehasonlítjuk alapszintű eredményeinkkel.

Teljesítménygondok beazonosítása

Amikor egy program teljesítményét megpróbáljuk növelni, az első lépés annak megtalálása, hogy mely része nem teljesít megfelelően. Erre a célra két eltérő módszert szoktam alkalmazni: a kódátnézést és a megfigyelést (profiling).

A teljesítménynövelési célú kódátnézés célja felderíteni a kódban található gyanús részeket. A kódátnézés előnye, hogy a vizsgálódó a programon belül figyelheti meg az adatok áramlását. Az adatok áramlásának megértése nagymértékben segítheti az eltávolítható vezérlési ciklusok felderítését, egyúttal megkönnyíti a később programmegfigyelés alá veendő kódrészletek beazonosítását. Nem javaslom, hogy a teljesítményvizsgálati kódátnézést egyéb célú, például szabványossági kódátnézéssel keverjük.

A programmegfigyelés folyamata során a program futását vizsgáljuk, hogy megállapítsuk, hol tölt el nagyobb időt, illetve hogy az egyes műveleteket milyen gyakran hajtja végre. Erre a célra a `Benchmark::Timer` Perl csomagot használtam; a csomagban található függvények segítségével megjelölhető a kérdéses kódrészlet eleje és vége. Az összes ilyen megjelölt kódrészletet egy-egy címke azonosítja. Amikor a program futtatása során egy megjelölt részletbe lépünk, a kérdéses szakasz által felhasznált időmennyiség rögzítésre kerül.

A programunkba illesztett megfigyelőrészek módszere eléggé erőszakos; ugyanis megváltoztatja a kód viselkedését. Más szóval, könnyen előfordulhat, hogy a megfigyelő kód elfed vagy meghamisít valamilyen teljesítménygondot (Performance Problems). A teljesítménynövelés korai szakaszában ez nem feltétlenül okoz nehézséget, hiszen a teljesítmény problémája nagyságrendileg nagyobb, mint a megfigyelő kód hatása a teljesítményre. A teljesítménygondok kiküszöbölése után azonban egyre valószínűbb, hogy a további teljesítménykérdéseket már nehezebb lesz megkülönböztetni. Mint sok más dolog, a teljesítmény növelése is iteratív folyamat.

Esetünkben néhány rész megfigyelése azt mutatta, hogy jelentős idő fordítódik a gépekről begyűjtött adatok statisztikáinak elkészítésére. Átnéztem a statisztikák készítéséhez kapcsolódó kódokat, és kiderült, hogy igen gyakran használjuk a normálszórást előállító `std_dev` függvényt.

A `std_dev` két okból is szemet szúrt: először is azért, mert a normálszórás számításához a teljes mérési készlet átlagát és négyzetátlagát is ki kell számítani. Az `std_dev` függvényhez használt számítások két ciklust használnak, holott egy is elegendő lenne. Másodszor észrevettem, hogy a vermen keresztül – egyszerű hivatkozás helyett – a teljes adatkészletet átadjuk az `std_dev` függvénynek. Úgy gondoltam, hogy ennek a két dolognak a hatása a teljesítményre már megéri a vizsgálódást.

1. lista A std_dev alapmegoldása

```
sub mean {
  my $result;
  foreach (@_) { $result += $_ }
  return $result / @_;
}


sub std_dev {
  my $mean = mean(@_);
  my @elem_squared;
  foreach (@_) {
    push (@elem_squared, ($_ **2));
  }
  return sqrt( mean(@elem_squared) -
              ➥($mean ** 2));
}
```

Teljesítménymérés

Miután beazonosítottuk a fejlesztésre szoruló függvényt, áttértem a következő lépésre, a teljesítménymérésre. A teljesítménymérés során összehasonlítási alapként megállapítjuk a kezdeti teljesítményt. Kizárólag a teljesítménypróba segítségével tudhatjuk meg, hogy változtatásaink javítottak-e bármit is a teljesítményen. Az itt bemutatott valamennyi teljesítménypróba időalapú. Szerencsére létezik egy külön időalapú teljesítménymérésre kifejlesztett Perl-csomag, a Benchmark.

A `std_dev` függvényt (*1. lista*) a programból átmásoltam a próbaparancsfájlba. Azzal, hogy a kérdéses részt a próbarendszerbe másoltam, az adatgyűjtő program hatásai nélkül mérhetem a teljesítményt. Mivel mérvadó teljesítményadatokat szeretnék kapni, az adatgyűjtő program körülményeihez hasonló terhelést kell előállítanom.

Az adatgyűjtő program által feldolgozott adatokat megvizsgálva úgy találtam, hogy 0 és 999,999 közötti számok keveréke megfelelő lesz. Ahhoz, hogy helyes teljesítmény eredményt kapjunk, az `std_dev` függvényt többször meg kell ismételnünk. Minél több alkalommal hajtódik végre a függvény, annál megbízhatóbb és egységesebb lesz a teljesítménypróba. A Perl Benchmark csomagban beállíthatjuk, hogy hányszor szeretnénk megismételni a próbát. Példaképpen tesztünket 10 000 alkalommal futtassuk. Másik lehetőségként egy adott időintervallumot is megadhatunk, ilyenkor a próbaprogram a megadott időn belül annyiszor futtatja a függvényt, ahányszor csak bírja.

A próbában bemutatott valamennyi teljesítménypróba 10 másodperces időintervallumot használ. 1 000 000 adatelem normálszórásának legalább 10 másodpercig tartó kiszámítása a következő eredményt adta:

```
12 wallclock secs (10.57 usr + 0.02 sys =
➥10.59 CPU) @ 0.28/s (n = 3)
```

A fenti sorokból megtudhatjuk, hogy a teljesítménymérés 12 másodpercig tartott. A teljesítménymérő eszköz másodpercenként 0,28 alkalommal tudta a függvényt futtatni, azaz – megfordítva – egy ciklus 3,5 másodpercig tartott. A teljesítménymérő eszköz a megadott 10 processzor-másodperc alatt mindössze három alkalommal tudta futtatni a függvényt (n = 3). A továbbiakban cikkünkben a teljesítményeredményeket másodperc/iteráció mértékegységben közöljük (s/iter). Minél alacsonyabb ez a szám, annál nagyobb a teljesítmény. Példaképpen egy végtelen gyors függvény 0 s/iter értéket adna, egy nagyon rossz függvény mondjuk 60 s/iter értéket. Az `std_dev` alapteljesítményének megállapítását követően már meg lehet állapítani a függvény újraírásának a hatását.

Az `std_dev` számítással kapcsolatos nehézségek feltárásához ugyan három ciklus is elegendő, mélyebb teljesítményelemzést azonban csak további minták alapján érdemes végezni.

Újraírás és ellenőrzés

Az első listában közölt teljesítményteszt végrehajtása után két lépésben újraírtam az `std_dev` algoritmust. Az első, `std_dev_ref` nevű változatban az értékátadás módszerét változtattam meg, „érték szerinti átadás”-ról „cím szerinti átadás”-ra az `std_dev` függvényben és az `std_dev` által meghívott átlagfüggvényben. Az eredményül kapott függvények a *2. listában* találhatók. Ez a változtatás elméletileg mindkét függvény esetében növeli a teljesítményt, hiszen a programnak az `std_dev`, majd az azt követő átlagfüggvény meghívása előtt nem szükséges a teljes adatállományt a verembe másolnia.

2. lista Érték szerinti átadás helyettesítése cím szerinti átadással

```
sub mean_ref {
  my $result;
  my $ar = shift;
  foreach (@$ar) { $result += $_ }
  return $result / scalar(@$ar);
}

sub std_dev_ref {
  my $ar = shift;
  my $mean = mean_ref($ar);
  my @elem_squared;
  foreach (@$ar) {
    push (@elem_squared, ($_ **2));
  }
  return sqrt( mean_ref(\@elem_squared) -
               ($mean ** 2));
}
```

3. lista Az átlagfüggvény eltávolítása után

```
sub std_dev_ref_sum {
  my $ar = shift;
  my $elements = scalar @$ar;
  my $sum = 0;
  my $sumsq = 0;

  foreach (@$ar) {
    $sum += $_;
    $sumsq += ($_ **2);
  }
  return sqrt( $sumsq/$elements -
               (($sum/$elements) ** 2));
}
```

A második módosításban, amelynek a `std_dev_ref_sum` nevet adtam, az átlagfüggvényt egy az egyben eltávolítottam. Az átlag és a négyzetátlag számítását a teljes adatkészletre vonatkoztatott egyetlen ciklusba helyeztem át. Ez a *3. listában* olvasható módosítás legalább két, az összes adaton végigfutó ciklust távolít el. Az *1. táblázat* a teljesítménymérési időeredményeket foglalja össze. Ahogy reméltük, az 1. táblázatban minden egyes lépésnél teljesítménynövekedést tapasztalhatunk. Az `std_dev` és `std_dev_ref` függvények közt 20 százalék növekedés

1. táblázat *Az alap és két módosítás*

	másodperc/iteráció
std_dev	3.53
std_dev_ref	2.93
std_dev_ref_sum	1.37

4. lista Az std_dev_pm függvény

```
sub std_dev_pm {
  my $stat = new
     ➥Statistics::Descriptive::Sparse();
  $stat->add_data(@_);
  return $stat->standard_deviation();
}
```

látszik, a `std_dev` és `std_dev_ref_sum` függvények közt pedig 158 százalék növekedést sikerült elérni. Ez bizonyítani látszik azt a feltevésemet, hogy a cím szerinti átadás Perlben gyorsabb, mint az érték szerinti (Nagy adatkészletekre igaz, néhány bájtos átadásnál nem feltétlenül – a ford.). Hasonlóképpen az adatokat kezelő két ciklus eltávolítása ismét növelte az `std_dev_ref_sum` függvény teljesítményét. A két módosítást követően a függvény 1,37 másodperc alatt képes kiszámítani 1 000 000 elem normálszórását. Igaz ugyan, hogy ez már lényegesen jobb, mint az eredeti, de azért azt hiszem, még mindig lehet javítani rajta.

Nem készítette el már valaki?

Rengeteg nyílt forrású Perl-csomag érhető el. Szerencsés esetben ráakadhatok egy olyan normálszórás-függvényre, amely gyorsabb, mint az eddigi legjobb próbálkozásom. A CPAN-on valóban rá is akadtam a `Statistics::Descriptive` nevű csomagra, amit letöltöttem. Létrehoztam egy `Statistics::Descriptive` csomagot használó `std_dev_pm` nevű függvényt; a függvényhez tartozó programot a *4. listában* találjuk. A függvény azonban csak 6,80 s/iter teljesítményt ért el, ami 48 százalékkal rosszabb, mint az alapul szolgáló `std_dev` függvényé. Ha belegondolunk, hogy a `Statistics::Descriptive` csomag objektum csatoló-felületet használ, ez igazából nem is meglepő eredmény. Minden számításban többletmunkát jelent ugyanis a `Statistics::Descriptive::Sparse` objektumok létrehozása és törlése. Ez nem azt jelenti, hogy a `Statistics::Descriptive` rossz csomag lenne. Figyelemre méltó mennyiségű Perlben írt statisztikai függvényt tartalmaz és nagyon könnyen használható olyan számításokra, amelyeknél a sebesség nem döntő tényező. Csakhogy a mi esetünkben éppen a sebesség a fontosabb.

Egy nyelven kívüli élmény

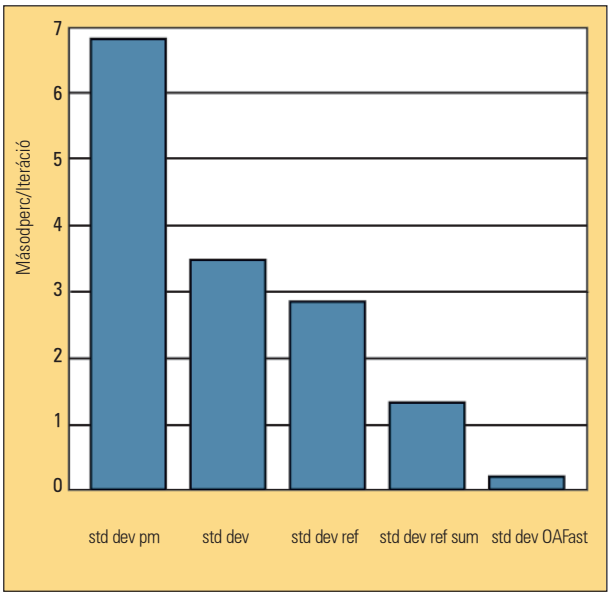
Minden nyelvnek van jó és rossz oldala. A Perl például kiváló általános célú nyelv, de tömeges számfeldolgo-

5. lista Az XS alapú megoldás

```
double
std_dev(sv)
    INPUT:
        SV *            sv
    CODE:
        double sum = 0;
        double sumsq = 0;
        double mean = 0;

        /* Dereference a scalar to retrieve
           an array value */
        AV* data = (AV*)SvRV(sv);

        /* Determine the length of the array */
        I32 arrayLen = av_len(data);

        if(arrayLen > 0)
        {
          for(I32 i = 0; i <= arrayLen; i++)
            {
              /* Fetch the scalar located at i
                 from the array.*/
              SV** pvalue = av_fetch(data,i,0);

              /* Dereference the scalar into
                 a numeric value. */
              double value = SvNV(*pvalue);

              /* collect the sum and the
                 sum of squares. */
              sum += value;
              sumsq += value * value;

            }
          mean = (sum/(arrayLen+1));
          RETVAL = sqrt((sumsq/(arrayLen+1)) -
                        (mean * mean));
        }
        else
        {
          RETVAL = 0;
        }

    OUTPUT:
        RETVAL
```

2. táblázat *Az alap és a leggyorsabb Perl-megoldás összehasonlítása a C-ével*

	másodperc/iteráció
std_dev	3.53
std_dev_ref_sum	1.37
std_dev_OAFast	0.277



Valamennyi megoldás összehasonlítása

zásra nem a legjobb. Ezt szem előtt tartva úgy döntöttem, hogy a normálszórás-függvényt C nyelven újraírom, hogy lássam, növeli-e a teljesítményt.

Az adatgyűjtő alkalmazás esetében nem lett volna kifizetődő a teljes programnak C nyelven történő újraírása. A legtöbb alkalmazás elkészítésére számos különleges függvénye a Perlt teszi a legalkalmasabb nyelvvé. Az alkalmazás újraírásának legjobb módszere, ha csak a teljesítmény növeléséhez szükséges függvényeket írjuk újra. Ezt a C nyelven megírt normálszórás-függvény Perl-modulba burkolásával érjük el. A C-függvény beburkolása révén a program nagyobbik része Perlben futhat, de a szükséges helyeken C- és C++-kódot is alkalmazhatunk.

A már létező C- vagy C++-csatolófelület fölé az XS használatával írhatunk Perl-burkolót (wrapper). Az XS eszköz a Perl-csomagban megtalálható, dokumentációját a ***perlxs*** Perl-dokumentumban találjuk. Szükségünk lesz néhány, a perlguts dokumentumban található információra is.

Az XS segítségével létrehoztam egy `OAFastStats` nevű Perl-csomagot, amely a C alapú normálszórás függvényt tartalmazza. Ezt az *5. listában* bemutatott függvényt azután Perlből már közvetlenül meg lehet hívni. Az összehasonlíthatóság kedvéért ezt a normálszórás függvényt `std_dev_OAFast` névre kereszteltem.

Az alap normálszórás- és a XS-el burkolt C alapú-függvény teljesítménye közti eltérést a *2. táblázat* mutatja be, miszerint jelentős sebességnövekedést tapasztalhatunk. A C-függvény (`std_dev_ref_OAFast`) 1175 százalékkal fürgébb, mint az alapul használt függvényünk (`std_dev`), és 395 százalékkal gyorsabb, mint a legjobb Perl-megoldás (`std_dev_ref_sum`).

Összegzés

A feladat során beazonosítottam azt a függvényt, amelyik valószínűleg nem teljesít olyan jól, mint kellene. A Perl nyelvű számítások újraírásával képes voltam bizonyos sebességnövekedést elérni. Kipróbáltam egy nyílt forrású csomagot, de kiderült, hogy 48 százalékkal rosszabb teljesítményt nyújt, mint az eredeti függvényem. Végül C nyelven elkészítettem a normálszórás függvényt, és az XS rétegen keresztül beépítettem a Perlbe. A C-változat az eredeti Perl-változathoz képest 1175 százalékos sebességnövekedést mutatott. Az egyes fejlesztéseket az *ábránkon* foglaltam össze.

A legtöbb esetben úgy tűnik, hogy a Perl teljesítménye képes versenyezni a C-vel – a mostani azonban nyilvánvalóan nem az az eset. A Perl kiváló általános célú nyelv, amelynek az egyik előnye éppen az, hogy képes átlépni a nyelv határait, és alacsonyabb szintű nyelvek kódjait beépíteni. Ne féljünk a nyelvek keverésétől, ha tényleg szükség van a teljesítmény növelésére; ám tudnunk kell, hogy ez karbantartási többletmunkát jelent. A további nyelvek bevezetésének az a hátulütője, hogy az alkalmazást később karbantartani kívánókra többletterhet ró: ismerniük kell a C nyelvet és érteniük kell az XS függvényt. Esetünkben azonban a megnövekedett teljesítmény lényegesen nagyobb súllyal esett latba, mint az XS támogatásával járó nehézségek.

Linux Journal 2004. február 116. szám

Bruce W. Lowther (blowther@micron.com) Idaho államban a boisei Micron Technology, Inc. programmérnöke. Kilenc éve dolgozik a Micronnál, ahol az utóbbi öt évben olyan eszközökön dolgozik, amelyek félvezető felszereléseket segítenek beépíteni a Micron gyártásfolyamatába.

Perl beágyazása MySQL rendszerbe

Készítsünk a MySQL-hez saját szolgáltatásokat a MyPerl segítségével, amelyek a hatékony és könnyen változtatható Perl-értelmező erejét hozzák el a relációs adatbázisok szívébe.

A MySQL igen gazdag függvénykészlettel rendelkezik, mégis könnyen előfordulhat, hogy egyedi igényeink támadnak, esetleg fejlettebb szabályoskifejezés-kezelő motorra van szükségünk. Az ilyen gondok megoldására találták ki MySQL alatt a felhasználói függvényeket (user-defined functions, UDF). Az UDF csatolófelületen keresztül dinamikusan új függvényeket tölthetünk be az adatbázisba. Annak ellenére, hogy igen hatékony módszerről van szó, most nem kell C- vagy C++-kódok hibáinak a javításával bajlódnunk. Bármennyire is kedvelem a C nyelvet, néha egyszerűen nincs időm az alkalmazások elkészítésére és a hibáik kijavítgatására, máskor pedig egyszerűen csak gyorsabb fejlesztési ciklust szeretnék. És itt lép be képbe a Perl, a nyelvek mindenese – ezer köszönet a Comprehensive Perl Archive Network (CPAN) hálózatának! A Perl MySQL környezetbe illesztése elképesztő rugalmasságot nyújtott adatbázisom gyors bővítéséhez. A fentieknek köszönhetően jött létre a MySQL Perl-értelmezője, a MyPerl.

A Perl beállítása

Az első lépés, amit az adatbázisba illesztés előtt meg kell tennünk, a Perl helyes beállítása lesz. A Perl ugyanis alapértelmezés szerint a szálakat (threads) nem kezeli, a MySQL viszont minden felhasználói kapcsolathoz külön szálat használ. Így ha az adatbázisunk belsejében Perlt szeretnénk üzemeltetni, kénytelenek leszünk Perl-száltámogatással lefordítani:

```
./Configure -Dusethreads -Duseithreads
```

Amint ezzel elkészülünk, máris egy száltámogatással rendelkező Perl-rendszert birtoklunk. Ez nem azt jelenti, hogy a kódunk vagy az abban felhasznált Perl-modulok szálbiztosak lesznek, mindössze annyit tudhatunk, hogy maga a Perl az lesz. A szálbiztos Perl fordítása elengedhetetlen, mivel egyetlen terjesztőt sem ismerek, aki szálbiztos Perl-változatot alkalmazna. Ne tévesszen meg bennünket, hogy a MyPerl nem szálbiztos Perl alatt is hajlandó lefordulni; lefordul ugyan, de a későbbiek során egyszer csak kifagyasztja az adatbázist. Gyanítom, hogy az Apache 2.0 és a `mod_perl2` eljövetelével néhány terjesztő már számításba fogja venni a szálak engedélyezésével fordított Perl-binárisok forgalmazását. Mostanában, miközben a cikk befejezéséhez készülődtem, Red Hat 9-re frissítettem, és láttam, hogy ezt a változatot már szálkezelést támogató Perllel szállították.

Beágyazott Perl-értelmező készítése

Az UDF-eknek három állapota van: `init` (inicializálás), `request` (kérelem) és `deinit` (deinicializálás). Az `init` lépcsőfok a lekérdezés elején, egyszer hívódik meg; a `request` állapot minden egyes soron végrehajtódik, végül a `deinit` állapot az adat ügyfélhez küldése után hívódik meg. Az `init` és `deinit` állapotokat átugorhatjuk, igaz, a legegyszerűbb UDF-ektől eltekintve szinte mindig szükségünk lesz memóriafoglalásra és- felszabadításra, hiszen így tudunk az ügyfélnek adatokat átadni.
A MyPerl a következő `init` függvénnyel kezdődik:

```
my_bool
myperl_init(UDF_INIT *initid, UDF_ARGS *args,
            ➥char *message)
{
     myperl_passable *pass = NULL;
     int exitstatus = 0;
     char *embedding[] = { "perl",
                           ➥"-MMyPerl::Request",
                           ➥"-e", "1" };
     PerlInterpreter *my_perl;
     uint i =0;
     initid->max_length = 256;
     initid->maybe_null=1;
```

Az `init` függvény három értéket kap meg, és siker vagy kudarc értékkel tér vissza. Az `UDF_INIT` szerkezet az UDF viselkedésével kapcsolatos értékeket tárolja, egyúttal ez az egyetlen szerkezet, amely a három állapot közt átadódik. Először is közöljük a MySQL-lel, hogy az UDF a `VARCHAR` méretnél nagyobb adatot fog visszaadni. A kiszolgáló figyelmét felhívjuk, hogy `VARCHAR` méretnél nagyobbra kell számítania, így azt feltételezi, hogy `blog` típust kap vissza. Bár a MyPerl nem tudja, hogy valóban ez fog-e történni, ezen a ponton semmilyen módon nem tudhatjuk meg, hogy mennyi adatot akar majd visszaadni, így a legbiztosabb, ha a kiszolgálónak `blog` típust jelentünk. Ezután a `maybe_null` értéket 1-re állítjuk, hiszen mindig fennáll a lehetőség, hogy

NULL értékeket kell visszaadnunk. A MyPerl egyaránt NULL értéket ad vissza, ha az eredmény üres, illetve, ha az eval() függvénnyel futtatott kód fordítási hibát jelez.
A következő feladat az átadandó sorok vizsgálata:

```
if (args->arg_count == 0 || i
     args->arg_type[0] != STRING_RESULT) {
     strncpy(message,USAGE, MYSQL_ERRMSG_SIZE);
     return 1;
}
for (i=0 ; i < args->arg_count; i++)
          args->arg_type[i]=STRING_RESULT;
```

A MyPerl elsőként a futtatandó kódot várja. Ezért aztán, ha egyetlen sort sem adunk át vagy ha az első sor nem string típusú, hibajelzést kapunk. A hibaüzenetek mérete legfeljebb MYSQL_ERRMSG_SIZE méretű lehet, amelyeket a message string változóba kell másolnunk. Időmegtakarításképpen a MyPerl végignézi az átadásra kerülő értékeket, és a MySQL kéri, hogy alakítsa őket karaktersorozattá.
A Perl-értelmező indítása előtt létre kell hoznunk egy szerkezetet, amelyben az értelmezőt tárolni fogjuk, illetve nyomon kell követnünk azt a memóriablokkot, ahol az egyes kérelmek visszatérő adatai kerülnek:

```
pass = (myperl_passable *)
     malloc(sizeof(myperl_passable));;
if (!pass) {
     strncpy(message, "Could not allocate memory",
             ➥MYSQL_ERRMSG_SIZE);
     return 1;
}
```

A szerkezet megadása:

```
typedef struct {
     char *returnable;
     size_t  size;
     PerlInterpreter *myperl;
     size_t  messagesize;
} myperl_passable;
```

A returnable karaktermutatót használjuk a memóriablokk címének tárolására, a size és messagesize változókat pedig a visszatérítendő adat jelenlegi és teljes méretének nyilvántartására. Minthogy a memória foglalása és felszabadítása igen erőforrás-igényes folyamat, nagyon fontos, hogy a lehető legkevesebbszer éljünk vele. A Perl-értelmezőt szintén ebben a szerkezetben tároljuk.
A lekérdezéshez használható Perl-értelmező beállításával lényegében el is készültünk. Jelenleg a MyPerl minden egyes lekérdezéshez külön Perl-értelmezőt készít, hogy elkerülje a memóriaszivárgást és ne veszélyeztesse a lekérdezések között az adatok biztonságát. Nagy valószínűséggel a jövőben inkább Perl-értelmezők egy csoportját fogjuk alkalmazni:

```
if((my_perl = perl_alloc()) == NULL) {
     strncpy(message, "Could not allocate perl",
             ➥MYSQL_ERRMSG_SIZE);
     return 1;
}
perl_construct(my_perl);
exitstatus = perl_parse(my_perl, xs_init, 4,
                        ➥embedding, environ);
PL_exit_flags |= PERL_EXIT_DESTRUCT_END;
if (exitstatus) {
     strncpy(message, "Error in creating perl
             ➥parser",
              MYSQL_ERRMSG_SIZE);
     goto error;
}
exitstatus = perl_run(my_perl);
if (exitstatus) {
     strncpy(message, "Error in parsing your perl",
             ➥MYSQL_ERRMSG_SIZE);
     goto error;
}
```

Az első függvény, a perl_alloc(), új Perl-értelmezőt foglal le, amelyet aztán a perl_construct() hoz létre. Innentől már mindössze annyi dolgunk maradt, hogy lefuttassuk a Perlt. A beágyazott változót értékként használhatjuk a Perl-értelmezőhöz – ezek pontosan ugyanazok az értékek lesznek, amelyeket a parancssorban is alkalmaznánk. Valahányszor a Perl-értelmezőhöz nyúlunk, ellenőriznünk kell a hibákat. A jelenlegi megoldásban hiba esetén a MyPerl olyan függvényekre ugrik, amelyek felszabadítják a lefoglalt memóriát.
Sikerült előállítanunk egy használható Perl-értelmezőt, ideje, hogy az átadási szerkezetben is beállítsunk néhány alapértelmezett értéket. Az értelmezőt be kell állítanunk, a szerkezet

címét pedig az `initid->ptr` mutatóban kell tárolnunk, hogy aztán a teljes lekérdezés ideje alatt használható legyen:

```
pass->returnable = NULL;
pass->size = 0;
pass->messagesize = 0;
pass->myperl = my_perl;
initid->ptr = (char*)pass;
return 0;
```

A fenti beállítások után a MyPerl már készen áll a lekérdezések fogadására:

```
char *
myperl(UDF_INIT *initid, UDF_ARGS *args,
       ➥char *result, unsigned long *length,
       ➥char *is_null, char *error)
{
    myperl_passable *pass =
        (myperl_passable *)initid->ptr ;
    char *returnable = NULL;
    unsigned long x = 0;
    size_t size = 0;
    char *newspot = NULL;
    char *string = NULL;
    myperl_passable *pass =
         ➥(myperl_passable *)initid->ptr ;
    STRLEN n_a;
    PerlInterpreter *my_perl = pass->myperl;
```

Nagyon fontos, hogy az értelmező nevét a `my_perl` nevű változóba másoltuk. A Perl belső részek nagy többsége olyan makrókon alapul, amelyek elvárják, hogy változókat bizonyos néven nevezzünk. Az `STRLEN` változótípust a karaktersorozatok méretének tárolására használja. Meghívjuk az értelmezőt:

```
dSP;
ENTER;
SAVETMPS;
PUSHMARK(SP);
//a maradék értéket átadjuk az ARGV-nek
for(x = 0; x < args->arg_count  ; x++) {
    XPUSHs(sv_2mortal(newSVpvn(args->args[x],
          ➥args->lengths[x])));
}
PUTBACK;
call_pv("MyPerl::Request::handler", G_SCALAR);
SPAGAIN;
string = POPpx;
size = (size_t)n_a;
```

Az `XPUSHs` feladata az összes sort karaktersorozatok tömbjévé alakítani, amelyet aztán a `MyPerl::Request()` könyvtár `handler()` Perl függvénye kap meg. Ez a Perl-modul nagyon hasonló az `Apache::Request` moduljához, attól eltekintve, hogy ahol az Apache a fájlnevet használja a végrehajtott kód nyomon követéséhez, ott a MyPerl magát a kódot.
Minthogy a változó névleges mérete most már a visszatérítendő adatméretet is tartalmazza, le kell foglalnunk hozzá a helyet:

```
if (size) {
    if(pass->size < size) {
        newspot = (char *)realloc
                  ➥(pass->returnable, size);
        if(!newspot) {
            error[0] = ´1´;
            returnable =  NULL;
            goto error;
        }
        pass->size = size;
        pass->returnable = newspot;
    }
    // a jelenlegi méretet mindig tudjuk,
    // ez kevesebb is lehet, mint a teljes méret
    pass->messagesize = size;
    memcpy(pass->returnable, string, size);
} else {
    is_null[0] = ´1´;
}
error:
PUTBACK;
FREETMPS;
LEAVE;
*length = pass->messagesize;

return pass->returnable;
}
```

Itt tároljuk a kiszolgálónak küldendő adatokat. A `realloc()` memóriahívást használjuk, ha további memóriára van szükségünk. Ha a Perl-értelmezőtől nem kapunk további adatot, az `is_null` értékét 1-re állítjuk. Ezáltal a MySQL tudni fogja, hogy `NULL` értéket kell visszaadnia az ügyfélnek. A MyPerl azt is ellenőrzi, hogy a `call_pv()*` függvényben esetleg használt memóriát felszabadítottuk-e.
Ezután az összes sorra meghívjuk a `myperl()` függvényt. Miután a MySQL az ügyfélnek visszaadta az adatokat, meghívja a `deinit` függvényt, felszabadítva az értelmezőt és a további lefoglalt memóriát:

```
void myperl_deinit(UDF_INIT *initid)
{
    myperl_passable *pass =
          ➥(myperl_passable *)initid->ptr ;
    perl_destruct(pass->myperl);
    perl_free(pass->myperl);
    free(pass->returnable);
    free(initid->ptr);
}
```

Példák a MyPerl használatára

Most már birtokunkban van egy Perl UDF, így végre tudjuk hajtani a következő egyszerű trükköt:

```
mysql> select myperl(´return $ARGV[0]´,User)
       from mysql.user;
```

Mint láthatjuk, minden sor az `@ARGV` változó értékeinek

felel meg. A CPAN-modulokkal kiegészített MyPerlt használhatjuk közvetlen adatbevitelre is. A következő példa URL-ek listáját tölti le, majd a tartalmat adatbázisba helyezi:

```
mysql> insert into html select
       myperl("use LWP::Simple;
       my $content = get($ARGV[0]);
       return $content", url) from urls;
```

Az `XML::Simple` és `XML::XPath` modul segítségével bármilyen olyan XML formátumot lekérdezhetünk, amit esetleg az adatbázisunkban tároltunk. Én a MyPerl segítségével ellenőriztem például az adatbázisomban tárolt, sorosított (serialized) Perl-objektumokat.

Mi a helyzet a GROUP BY utasítással?

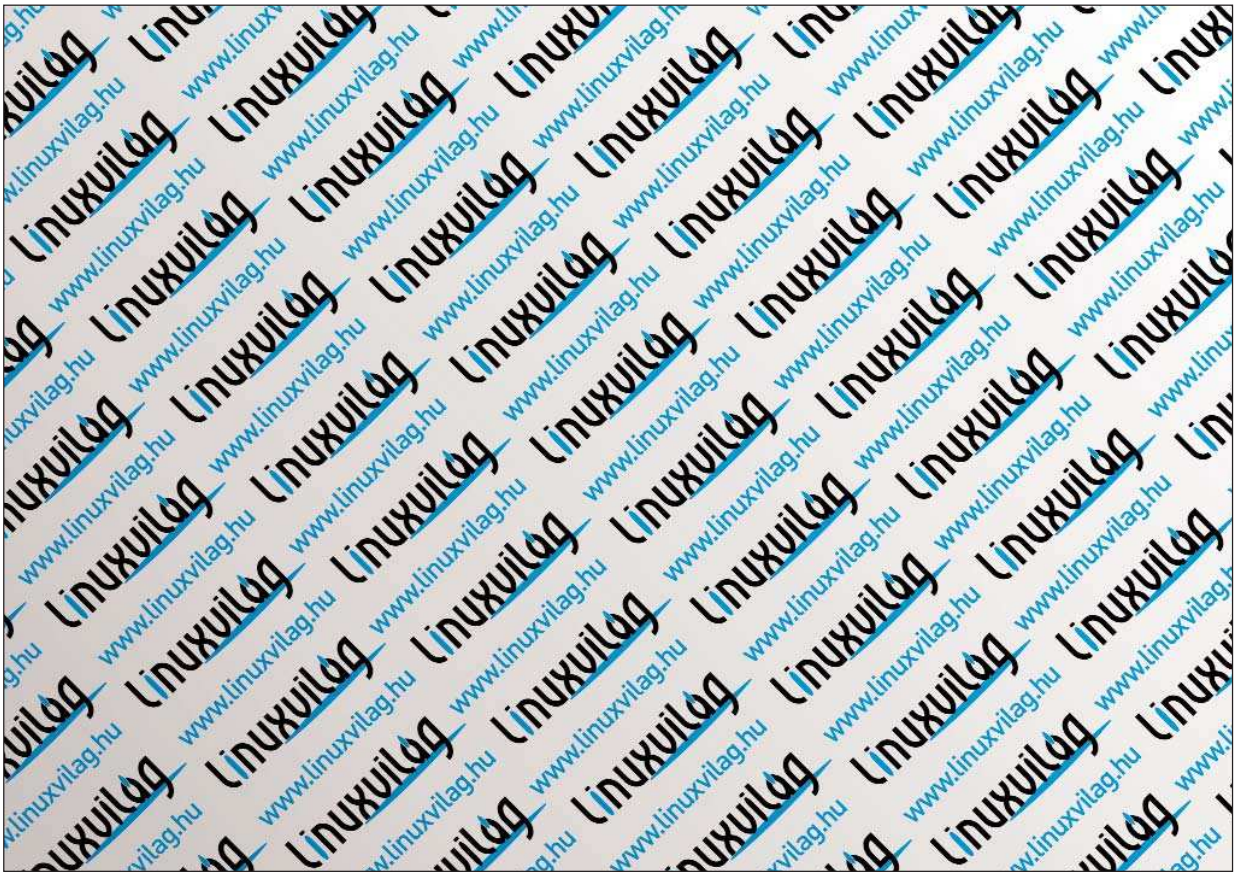
Fentebb ugyan bemutattuk, hogyan használjuk a kódot sorok lekérdezése esetén, mindez nemigen fog működni azoknál a lekérdezéseknél, amelyek `GROUP BY` segítségével kezeli az adatkészletet. Ezért létezik egy másik UDF-fajta, amelyet aggregátnak nevezünk. Az aggregátok abban különböznek szelídebb unokatestvéreiktől, hogy két további állapotuk is létezik: a `reset` és az `add`. Az aggregát UDF-ek esetében az add függvény kezeli az egyes sorokat, és a `request` lépcső válogatja ki az eredményeket és küldi el az adatokat az ügyfélnek. A `reset` állapot minden adatkészletnek az elején hívódik meg, ezért mindenképpen lefut, legalább egyszer. A MyPerl már most is rendelkezik aggregát UDF-fel, ez azonban még tervezés alatt áll.

Összegzés

Azáltal, hogy a Perlt beillesztjük a MySQL rendszerébe, az adatbázissal elvégezhető műveletek kínálata jócskán kiszélesedik. Igaz ugyan, hogy sok esetben éppen az a legjobb megoldás, ha a lehető legegyszerűbbre formáljuk az adatbázisunkat, vannak esetek, amikor ez mégsem tűnik célravezetőnek. Képzeljük el, hogy gigabájt nagyságrendű szövegeket kell az adatbázisból kivennünk és további feldolgozásra az ügyfélnek továbbküldenünk. Az adatküldésre fordított idő lesz a mérvadó; ha a Perl segítségével képesek vagyunk közvetlenül az adatbázisban feldolgozni az adatokat, bizony, nem kevés időt takaríthatunk meg. Ha használhatjuk a Perl fejlett szabályos kifejezéseit, könnyen készíthetünk ügyfeleket olyan nyelveken is, amelyek nem rendelkeznek kifinomult kifejezéstámogatással. Biztos vagyok benne, hogy a saját környezetünkben is jó alkalmazási lehetőségeket találunk. A ➲ http://software.tangent.org oldalon találjuk meg a MyPerlt, valamint további UDF-eket, amelyeket felhasználhatunk a saját változatunk elkészítéséhez.

Linux Journal 2003. december, 116. szám

Brian Aker (brian@tangent.org)
Ideje nagy részében MySQL- és Apache-modulokkal foglalkozik, keze nyomát viseli többek közt a mod_layout és a mod_mp3 Apache adatfolyam-szolgáltatásmodul is.



Alkalmazásaink kipróbálása az Xnee segítségével

Elfáradt a kattintóujjunk programunk grafikus felületének próbálgatásakor? Akkor foglaljuk héjba gonosz tesztjeinket az Xnee-vel!

Az Xnee a felhasználói tevékenység rögzítésére és ezek újrajátszására képes egy programon belül. Ha a program használatát rögzítjük vele, akkor az Xnee-vel később is kipróbálhatjuk azt. Ezeket a próbákat minden változat kiadása előtt (vagy akár minden este) elvégezhetjük programunk minőségének megőrzése érdekében. Valóban ilyen egyszerű lenne az egész? Majdnem.

Az Xnee nem csak grafikus felületek kipróbálására szolgál: a programot parancssorból indítható programjaink kipróbálására is használhatjuk. Ehhez néhány próbafájlt kell létrehoznunk, amelyek a parancssori alkalmazás összes kapcsolójának a működését ellenőrzik, majd a végén elemezhetjük az eredményeket. Az Xnee-t már annak vizsgálatára is használták, hogy mekkora adatforgalom bonyolódik le egy olyan nagyobb méretű hálózaton, amely számos, kis sávszélességgel rendelkező ügyfélgépből áll. Ha ugyanazokat a próbákat egyszerre több gépen is el szeretnénk végezni, akkor a program később kifejlesztett többgépes szolgáltatását használhatjuk. Az Xnee a programok kipróbálásán kívül azok bemutatására is alkalmas – ebben az esetben türelmes előadóként, fáradtság nélkül végzi a munkáját.

Az Xnee története

1997-ben ***Henric Johansson*** és jómagam PhD-dolgozatunk témájaként az X-események rögzítését és újrajátszását választottuk. Egy svéd vállalatnak készítettünk egy – nem ingyenes – felvevőt és újrajátszót, amit belső használatra kértek. Miután munkát találtam, szükségem volt egy ingyenes próbaprogramra X11 alá, tehát úgy döntöttem, hogy a dolgozatom körüli munka során összegyűlt tapasztalataimból merítve írok egyet. Az Xnee fejlesztése 1999 nyarán kezdődött és az elejétől fogva a GPL szerződés vonatkozott rá. 2002 novemberében jelent meg az 1.0-s változat és 2003 februárjára az Xnee GNU-csomag lett.

Bemutatás

Mielőtt belekezdenék az Xnee ismertetésébe, az alábbi rövid bevezetésben elmagyarázom a cikkben szereplő kifejezéseket. Az X egy grafikus-ablakos felhasználói felület többféle felületre; az X-kiszolgáló egy program, amely az eszközök kezelését és a képernyőre rajzolást végzi. A GNU/Linux és rendszereken az Xfree86 a leggyakrabban használt X-kiszolgáló. Az X-programokra ügyfél néven hivatkozunk, ilyen például az `xterm` vagy a Galeon. Az ügyfelek az X-protokoll segítségével érik el az X-kiszolgálót.

Ebben a cikkben az X-kiszolgáló és az ügyfelei közti adatcserére szolgáló csomagküldést vesszük górcső alá.

A csomagok nevei az alábbiak lehetnek: `Event` (esemény), `Request` (kérelem), `Reply` (válasz) és `Error` (hiba) – ezekre itt protokolladatként hivatkozunk. Az alábbi lista az X11-protokolladatok leírását tartalmazza:

- Az ügyfél kérelmet küld a kiszolgálóhoz, ami egy bizonyos feladat elvégzésére vagy adatok lekérésére vonatkozik.
- A kiszolgáló választ küld az ügyfélnek. Nem minden kérelmére érkezik válasz.
- A kiszolgáló az `Event` csomagot elküldi az ügyfélnek, ezzel értesíti arról, hogy felhasználói adatbevitel vagy valami olyasmi történt, amiről az ügyfélnek tudnia kell (például már nem az ő ablaka aktív).
- Ha a kérelem érvénytelen volt, a kiszolgáló az `Error` csomagot küldi el az ügyfélnek.

A legérdekesebb, hogy minden olyan esetben, amikor a felhasználó az egér vagy a billentyűzet segítségével kapcsolatba lép a számítógéppel, az X-kiszolgáló egy vagy több eseménycsomagot küld el az ügyfélnek. Ezek némelyike közvetlenül a felhasználói tevékenység hatására jön létre – az ilyen eseményeket eszközeseményeknek nevezzük. Ezek az alábbiak lehetnek: `ButtonPress` (az egérgomb lenyomása), `ButtonRelease` (az egérgomb elengedése), `MotionNotify` (az egérmutató mozgása), `KeyPress` (a billentyű lenyomása) és `KeyRelease` (a billentyű elengedése). Ha tehát mindezen eseményeket sikerülne rögzítenünk, akkor a felhasználói tevékenység minden részletét tárolhatnánk. Ha lenne egy olyan robotunk, amelyik képes lenne a tárolt adatokat értelmezi, majd az egérrel és a billentyűzettel újra lejátszaná az eseményeket, máris lenne egy próbarobotunk. Vagy ami még ennél is jobb: ha megoldhatnánk ezeknek az eseményeknek a „hamisítását” (programból történő előállítását), akkor programból megvalósított próbarobotról beszélhetnénk. Szerencsé-

re az X az események rögzítését és újrajátszását is támogatja. Az X-protokollon átvitt adatok rögzítéséhez a RECORD vagy az XTrap bővítményeket (plug-in) kell használnunk. A rögzítést más módon is megoldhatjuk, például az X-foglalat (socket) figyelésével, de most a RECORD-dal, az Xnee által is alkalmazott módszerrel foglalkozunk. A visszajátszás alatt a RECORD bővítmény hangolja össze a rögzített adatokat.
A RECORD az ügyfelek és a kiszolgáló között mozgó adatok másolatát elküldi az őket kérő ügyfélhez. A RECORD bővítménnyel tehát az Xnee az összes rögzíteni kívánt protokolladat fájlban való tárolására képes, hogy azután egy későbbi időpontban újrajátszhassa őket.
Az Xtest bővítmény az összes eszközeseményt hamisítani tudja, segítségével az Xnee egy élő felhasználó tevékenységét utánozhatja, például az egér mozgatását, a billentyűk vagy valamelyik egérgomb lenyomását és elengedését – más adat nem állítható elő vele.

Az Xnee telepítése

Az Xnee egy GNU-csomag, forráskódját az Xnee honlapján találjuk. Töltsük le a legfrissebb forrást, amely cikkünk írásakor az 1.0.6-os változat! Bontsuk ki, állítsuk be, építsük fel és telepítsük:

```
tar zxvf xnee-1.0.6.tar.gz
cd xnee-1.0.6
./configure
make
make install
```

A honlapról RPM csomagokat is letölthetünk, s az Xnee a FreeBSD ports tree-ben is elérhető. Az Xnee felhasználói és fejlesztői leírással érkezik, amelyek többféle formátumban is megtalálhatók a csomagban. A TeX formátumú kézikönyvek a programmal együtt érkeznek és a GNU FDL vonatkozik rájuk. Ha nem mi magunk szeretnénk a dokumentumokat felépíteni, akkor töltsük le őket valamilyen más formátumban (PDF, HTML, INFO, TXT) az Xnee honlapjáról. Jelen pillanatban az Xnee leírása az 1.0.4 változatnál jár. Bontsuk ki a fájlokat:

```
tar zxvf xnee-doc-1.0.4.tar.gz
```

Az Xnee futtatásakor ellenőrizzük, hogy a RECORD bővítmény engedélyezve van-e. XFree86 alatt ehhez a RECORD modult kell betöltenünk. Nyissuk meg az XFree86 beállítófájlt (ami általában a */etc/X11/XF86Config-4*), és nézzünk bele a Module részbe, ahol az alábbi sornak kell szerepelnie:

```
Load "record"
```

Erről az Xnee GYK-ban találunk további útmutatást.

Egyszerű példák a program használatára

Nem mélyedünk el azonnal az Xnee részletes taglalásában, előbb egy egyszerű példával kezdünk. Indítsuk el az Xnee-t a --all-events kapcsolóval, ami néhány esemény rögzítését állítja be. Ha komolyabban akarjuk használni a programot, akkor ez a kapcsoló kerülendő, de ebben a példában megfelelő lesz:

```
xnee --all-events
```

Amikor az egérmutatót vagy a billentyűket mozgatjuk, esetleg egérgombokat nyomunk meg, az Xnee adatokat ír ki az eseményről. Most egy egyszerű eseménysort rögzítünk, amelyet azonnal vissza is játszunk. Húsz egérmozdulat rögzítéséhez így indítsuk el az Xnee-t:

```
xnee --record --out session1.xnr
➥--device-event-range MotionNotify --loops 20
```

A parancssori kapcsolók és paraméterek jelentése: a rögzítés bekapcsolása (--record), az eredmények mentése a ***session1.xnr*** fájlba (--out session1.xnr), a MotionNotify események rögzítése (--device-event-range MotionNotify) és összesen húsz esemény tárolása (--loops 20).
Az eseménysor visszajátszásához az alábbi formát használjuk:

```
xnee --replay --file session1.xnr
```

A parancssori kapcsolók jelentése: a visszajátszás üzemmódot használjuk (--replay) és a ***session1.xnr*** fájlból vesszük az adatokat (***--file session1.xnr***).

A rögzítés határainak beállítása

Az Xnee-nek a határok beállításával magyarázhatjuk el, hogy mit kell rögzíteni. A határok kezdő (start) és záró (stop) értékkel rendelkeznek. Az alábbi adatokat rögzíthetjük: magkérelmek (core-request), eszközesemények (device-event), átvitt események (delivered-event), hibák (error), válaszok (reply), bővítési kérelmek (extension requests) és bővítési válaszok (extension reply). A cikkben nem elemezzük őket, aki bővebben szeretne erről a témáról olvasni, az nézze meg a RECORD bővítménydokumentumokat.
Lássunk azért néhány példát! A MotionNotify eszközesemény rögzítéséhez az alábbi kapcsolót használjuk:

```
--device-event-range MotionNotify
```

Ha a KeyPress-től a MotionNotify és CreateNotify eseményekig terjedő tartományt kívánjuk rögzíteni, akkor a következő kapcsoló használatára van szükségünk:

```
--device-event-range KeyPress, MotionNotify,
➥CreateNotify
```

Ha rövidebb parancssort szeretnénk, akkor az eseménynévhez tartozó számot is használhatjuk a nevek helyett. A rögzíteni kívánt adat számát az Xnee --print-data-name kapcsolójával kereshetjük meg:

```
xnee --print-data-name
```

Az Xnee leállítása

A munkát a rögzíteni kívánt adat számának beállításával (--loops kapcsoló) állíthatjuk le, vagy a TERM jel elküldésével (nyomjuk le a CTRL+C billentyűket abban a terminálablakban, ahonnan az Xnee-t indítottuk) megszakíthatjuk a program futását. Azt is megtehetjük, hogy egy billenytű-

1. táblázat **X-események a Galeon-próba kezdetekor**

A protokolladat neve	A felhasználó/ügyfél tevékenysége
MotionNotify	A felhasználó az egérmutatót a Galeon indítóikonjára viszi.
ButtonPress	A felhasználó megnyomja az egérgombot és a Galeon elindul.
CreateNotify	A Galeon elindul és az ablak megnyílik.
VisibilityNotify	A kezdőlap betöltődik és a felhasználó számára láthatóvá válik.
MotionNotify	A felhasználó a betöltött oldalon egy hivatkozás fölé viszi az egérmutatót.
ButtonPress	A felhasználó a hivatkozásra kattint.
VisibilityNotify	Az új oldal betöltődik és a felhasználó számára láthatóvá válik.

kapcsolatot hozunk létre, amelyet rögzítés közben semmi másra nem fogunk használni. Ehhez a `--stop-key` kapcsoló szükséges. Ha azt szeretnénk, hogy az Xnee a Ctrl+Alt+A billentyűk lenyomására hagyja abba a rögzítést, a parancsot egészítsük ki az alábbiakkal:

```
--stop-key Control+Alt,a
```

Összehangolás

Vajon miért foglalkoznánk az eszközeseményeken kívül más adatok rögzítésével, ha úgysem tudjuk visszajátszani őket? Az Xnee ezeket az adatokat az összehangoláshoz használja, és a dolgok itt kezdenek egy kissé bonyolulttá válni.

Tegyük fel, hogy a Galeon vagy más webböngésző használatát szeretnénk rögzíteni. Rögzítés közben minden nagyszerűen működik és a hálózat is kifogástalanul használható, amikor azonban visszajátszuk a Galeonban történteket, nem tudjuk elérni az internetet. Ha a Galeon nem képes betölteni az oldalt, akkor a visszajátszást érdemes addig folyamatban tartani, amíg a hálózat újra elérhetővé nem válik és az oldal betölthető nem lesz.

A többi adat rögzítésével a programmal való munkát hangolhatjuk össze. Ha például a weboldalnak a Galeon ablakában megjelenített adatokat rögzítjük, akkor visszajátszáskor ugyanezeknek az adatoknak a továbbítására kell várnunk. Ez biztosítja, hogy az oldal betöltődik, mielőtt az események visszajátszása elkezdődne. Példánkban az egyszerűség kedvéért sok X-protokolladatot kihagytunk (*1. táblázat*). Az egyszerű eseményhalmaz visszajátszásakor az Xnee ugyanezeket az eseményeket használja (*2. táblázat*).

Mit hangoljunk össze mivel?

Bár az összehangolásra szükség van, nehéz feladat a hozzá használni kívánt adatok megtalálása. Az Xnee ezt bővítményekkel oldja meg, amelyek azt határozzák meg, hogy bizonyos alkalmazásoknál mely eseményeket kell rögzíteni. Ezeknek a bővítményeknek a neve a kipróbálni kívánt alkalmazás nevére utal. Ha egy általunk írt webböngészőt szeretnénk kipróbálni, akkor jó ötlet a

2. táblázat **Visszajátszás**

A protokolladat neve	Az Xnee által végrehajtott művelet
MotionNotify	Az Xnee az egérmutatót a Galeon indítóikonjára viszi.
ButtonPress	Az Xnee megnyomja a gombot és a Galeon elindul.
CreateNotify	Az Xnee ennek az eseménynek a továbbítására vár. Amikor a `CreateNotify` eseményt megkapja, a fájlban tárolt soron következő eseményre tér rá.
VisibilityNotify	Az Xnee ennek az eseménynek a továbbítására vár. Mivel a hálózat nem él és az oldal nem tölthető be, ez az esemény nem kerül továbbításra. Az Xnee tovább vár, majd végül az esemény továbbítódik és az Xnee folytathatja a visszajátszást.
MotionNotify	Az Xnee a betöltött oldal egyik hivatkozása fölé állítja az egérmutatót.
ButtonPress	Az Xnee a hivatkozásra kattint.
VisibilityNotify	Az új oldal betöltődik és láthatóvá válik.

Galeon bővítmény használata. Néha viszont nem találunk a programunkhoz megfelelő bővítményt, így ránk hárul a feladat, hogy az összehangolni kívánt protokolladatokat megtaláljuk. A következő példa remélhetőleg könnyít a helyzeten: most a `gnumeric` nevű programhoz kellene a megfelelő adatokat megtalálnunk. Először is indítsuk el a `gnumeric`-et, majd egy terminálablakban az Xnee-t az alábbi kapcsolókkal:

```
xnee  --delivered-event-range EnterNotify-
➥MappingNotify --human-printout --loops 1000
```

Ennek hatására egy csomó érdektelen esemény tölti be a képernyőt, tehát állítsuk le az Xnee-t. A nem kívánt eseményeket a határok beállításával szűrjük ki:

```
xnee --delivered-event-range EnterNotify-
➥KeymapNotify,VisibilityNotify-MappingNotify
➥--human-printout  --loops -1
```

Így már sokkal jobb. Kezdjük meg a rögzítést az alábbi paranccsal:

```
xnee  --delivered-event-range EnterNotify-
➥KeymapNotify, VisibilityNotify-CirculateRequest,
➥SelectionClear-MappingNotify --loops 1000 --out
➥session1.xnr
```

Írjunk valamit a `gnumeric` táblázatba és a menükkel illesszük be az aznapi dátumot vagy más adatokat. Ha elkészültünk, váltsunk át a terminálablakra és a Ctrl+C billentyűkkel állítsuk le a rögzítést. Elérkezett az idő az események visszajátszására. A `gnumeric`-et hagyjuk pontosan olyan állapotban, mint a rögzítéskor

volt, és visszajátszási módban indítsuk el az Xnee-t:

```
xnee  --replay -f session1.xnr
```

Az Xnee visszajátszás közben néha megáll egy pillanatra. Ez akkor történik, ha a protokolladatok nem a rögzítés sorrendjében kerülnek visszajátszásra. Ilyenkor az Xnee a futást megállítja addig, amíg a várt adatok (a fájlból beolvasva) a kiszolgálótól meg nem érkeznek. Végül az előre beállított várakozási időtartam leteltekor az Xnee újra próbálkozik. Ha a rögzített adatforgalmat és a visszajátszáskor elküldött adatokat nem tudja összehangolni, akkor bedobja a törölközőt.

Az Xnee-ben a bővítmények segítségével a rögzítés tulajdonságait is beállíthatjuk. Ha megtaláltuk alkalmazásaink beállításait, egy bővítményfájlban mentsük őket. E fájl formája a parancssori kapcsolók használatára emlékeztet. Új bővítményt a legegyszerűbben egy régi másolásával, módosításával és másik néven történő mentésével hozhatunk létre. Az Xnee-csomagban több ügyfélhez is találhatunk bővítményeket. Ha van kedvünk, az általunk készített bővítményeket elküldhetjük az Xnee fejlesztőinek – ehhez az Xnee honlapján találhatunk útmutatásokat.

Ha a programunk új ablakokat nyit meg a felhasználói adatbevitelhez, gondoskodnunk kell arról, hogy ezek ugyanazon a helyen jelenjenek meg. Az Xnee minden eszközesemény koordinátáját programunk kezdőablakához képest határozza meg, nem pedig az éppen megnyitott ablakhoz képest.

A rögzítés megkönnyítése érdekében készítsünk parancsfájlokat, amelyek egy adott feladat szempontjai szerint indítják el az Xnee-t. A parancsfájlhoz programindítót vagy új menüelemet hozhatunk létre az ablakkezelő menüben.

Összegzés

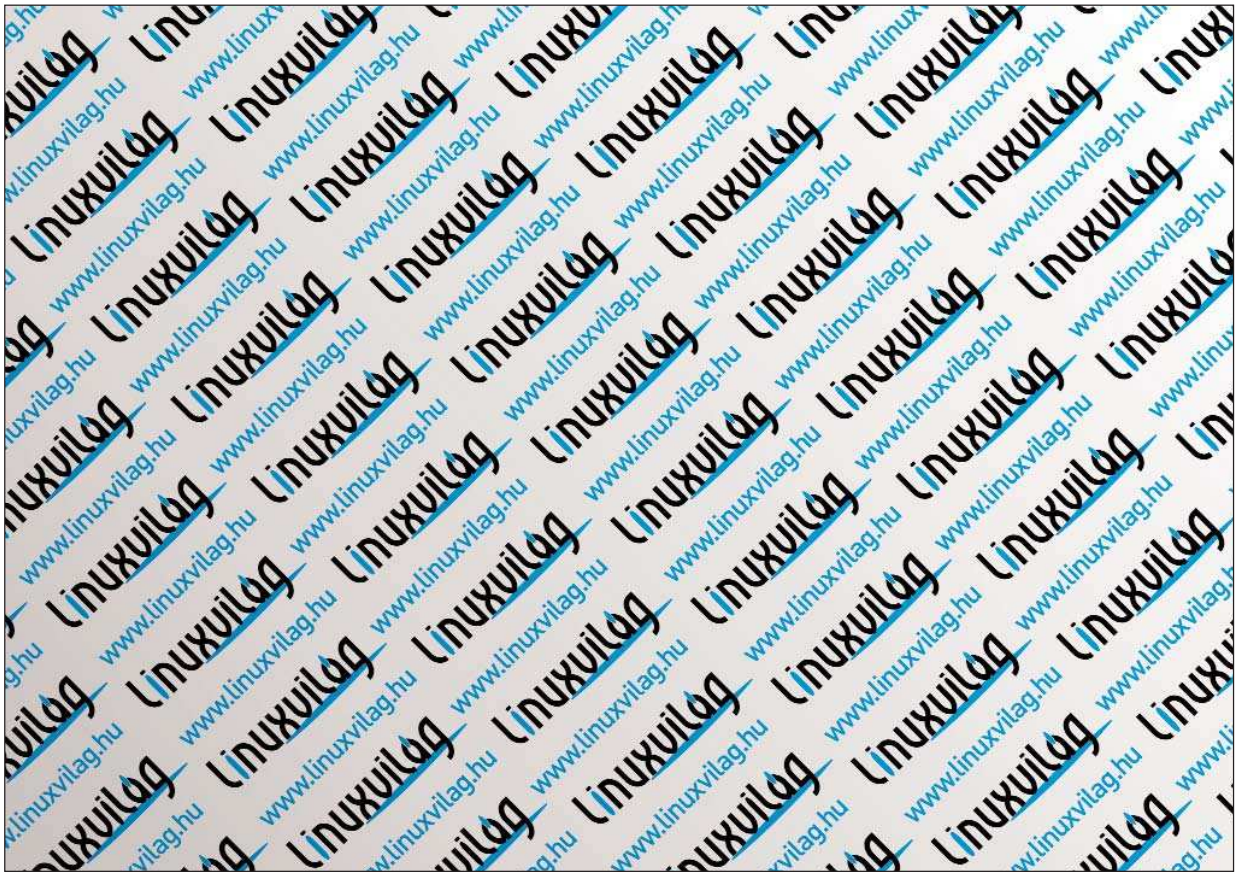
Az Xnee fejlesztése az utóbbi időben látványosan felgyorsult, javarészt a felhasználók hozzájárulásának köszönhetően. Remélem, hogy egyre többen választják az Xnee-t mindennapi társként a programfejlesztésben. Kellemes próbálgatást!

Linux Journal 2004. január, 117. szám

Henrik Sandklef (hesa@gnu.org)
Feleségével és lányaival a svédországi Gothenburgban él. A legtöbb idejét a családjával, főzéssel, GNU-programok boncolgatásával és kipróbálásával tölti, emellett alkalmanként a labdarúgással is megpróbálkozik.

KAPCSOLÓDÓ CÍMEK

GNU ➲ http://www.gnu.org
Kenton Lee ➲ http://www.rahul.net/kenton/xsites.html
XFree86 ➲ http://www.xfree86.org
Xnee ➲ http://www.gnu.org/software/xnee/www/index.html



A gyorstárazás rejtelmei

A Linux futtatására használt géptípusok jelentős eltéréseket mutatnak a gyorstárazás hardveres kezelésének mikéntjében. Tekintsük át, hogyan próbálja meg a rendszermag mindegyik gyorstármegoldásból a legjobbat kihozni.

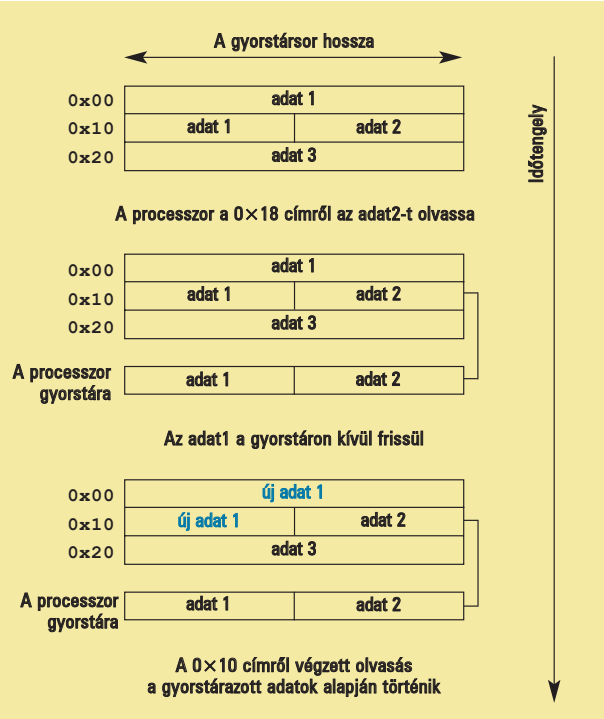
A tervezők számára az első mikroprocesszorok megjelenése óta folyamatosan gondot jelent, hogy a processzorok működési sebessége meghaladja az őket kiszolgáló memória-alrendszer sebességét. El szerették volna kerülni, hogy a processzorok a memóriából lekért adatokra ciklusokon keresztül tétlenül várjanak, ezért általános megoldásként egy gyorsabb – és drágább – memóriaterületet különítettek el a főmemória adatainak gyorstárazására. Így vált lehetővé, hogy a processzor saját képességeit maradéktalanul kihasználva üzemeljen, feltéve, hogy az ehhez szükséges adatok a gyorstárban rendelkezésre állnak.
Írásom célja, hogy a gyorstárazást rendszermag-programozási szempontból tekintsem át. Szó lesz néhány általánosan elterjedt, a gyorstárak kapcsán használt fogalomról, kifejezésről is. Írásom több részre ágazik, ezek egy része szorosabban kapcsolódik a programozás területéhez, míg mások inkább a gyorstárral foglalkoznak, mondanivalójuk a rendszermag működésétől független.

A gyorstár és jellemzői

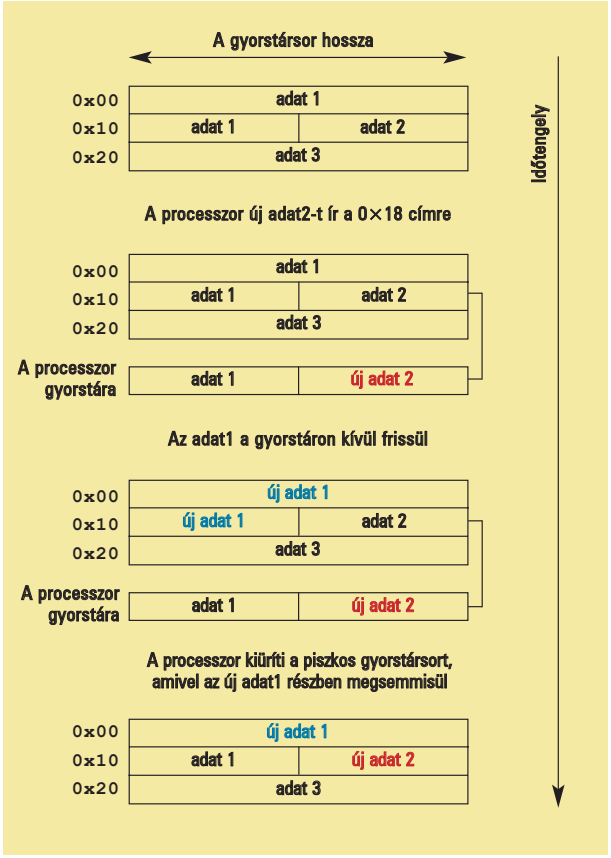
A gyorstár röviden és tömören egy olyan tárhely, amely átmenetitárazza (buffer) a memória-hozzáféréseket és szerencsés esetben rendelkezik a kért adatok másolatával. A gyorstárat, illetve – mivel több is lehet belőlük -tárakat általában úgy képzeljük el, mint egymásra tornyozott rétegeket. Legfelül a processzor, legalul a főmemória található, közöttük pedig egy vagy több gyorstár helyezkedik el. Ebben a rendszerben a gyorstárakat szintjük sorszámával azonosítjuk. A processzorhoz legközelebb lévő gyorstárat első szintűnek nevezzük (angolul Level 1, röviden L1), a további szintszámok a főmemória felé haladva növekednek.
A gyorstárba beírható és belőle kiolvasható legkisebb adategység a gyorstársor (cache line). Egy gyorstár egyik legfontosabb jellemzője az írási és olvasási sorok hossza – ez határozza meg, hogy a főmemóriából vagy az alsóbb szintű gyorstárból hány adatot kell egyszerre kiolvasni, illetve az ellenkező irányba haladva kiírni. A két érték gyakran megegyezik, így a gyorstárakat sokszor egyszerűen csak egyetlen sorhosszal jellemezzük. Ha a két érték eltér, akkor általában a nagyobbat szokás megadni.
A gyorstár másik fontos tulajdonsága a mérete. Ez adja meg, hogy hány adatot lehet benne tárolni. Ökölszabály, hogy minél nagyobb a gyorstár, annál jobb teljesítményre számíthatunk rendszerszinten.
Ha több gyorstárszint is van, ezek befoglalók vagy kizárók lehetnek. A kizárás azt jelenti, hogy egy-egy gyorstársor pontosan csak egy szinten szerepelhet. A befoglaló gyorstár megengedi, hogy egy-egy sorból egynél több szinten is legyen másolat. A sorhossz a különböző gyorstárszinteknél akár eltérő is lehet.
Végül egy gyorstár keresztülírásos (write through) vagy visszaírásos (write back) lehet. A keresztülírás azt jelenti, hogy a gyorstár tárolhatja az adat másolatát, de az írási műveletet az adott szinten csak akkor tekintjük befejezettnek, ha a kiírás az alatta lévő szinten is megtörtént. A visszaírásnál az írási művelet azonnal befejezettnek tekinthető, mihelyt az adatok bekerültek a gyorstárba. A visszaírásos gyorstáraknál a módosított sort addig piszkosnak (dirty) nevezzük, amíg az alsóbb szintű írás meg nem történik – természetesen ezt a kiírást előbb-utóbb mindenképpen meg kell ejteni.

Gyorstárkezelés és egységesség

A gyorstárak kezelése kapcsán az egyik legnagyobb gondot az egységesség, a koherencia biztosítása jelenti. A gyorstár egy sorát egységesnek tekintjük, ha a tartalma megegyezik az általa gyorstárazott főmemóriabeli adatokkal. Ha a két memóriatartalom különbözik, akkor ellentmondásosnak, inkoherensnek nevezzük őket. Az ellentmondásosság kétféle hibához vezethet; az első bármilyen gyorstárnál előállhat, ez az adatok elavulása. Ilyesmi akkor fordul elő, ha a főmemóriában tárolt adatok megváltoztak, de a gyorstár nem frissült, ezért a tartalma nem tükrözi a változásokat. Az eltérés általában hibás olvasást okoz, ennek folyamata látható az *1. ábrán*. A hiba könnyen elhárítható, hiszen a helyes adatok megtalálhatók a főmemóriában, mindössze át kell őket másolni a gyorstárba.
A második hiba csak visszaírásos gyorstáraknál állhat elő, ám ez sokkal súlyosabb, mivel adatvesztéshez vezethet. A folyamat a *2. ábrán* követhető nyomon: az adatok megváltoztak a főmemóriában, majd egy külön írási művelettel a processzor a gyorstár tartalmát is módosította. Mivel a gyorstár csak egy-egy sornyi adatot képes mozgatni, a két



1. ábra Az adatok elavulása



2. ábra Az adatok megsemmisülése a piszkos gyorstársorok miatt

változtatás nem hozható összhangba: vagy kiírás nélkül törölni kell a gyorstársort, ami a processzor által végrehajtott módosítás elvesztését okozza, vagy ki kell írni a gyorstár adatait, amivel viszont a főmemóriában elvégzett módosítás lesz az enyészeté. Minden programozónak törekednie kell az adatvesztéshez vezető helyzetek kialakulásának elkerülésére, ezt pedig a gyorstárkezelő hívások átgondolt alkalmazásával érhetik el.

A gyorstársorok ütközése

Ha két egymástól független adategység található ugyanabban a gyorstársorban, akkor sorütközésről beszélünk. Ilyenkor állhat elő az imént leírt adatvesztés lehetősége. Ha el akarjuk kerülni az ilyen helyzeteket, akkor az adatszerkezetek elhelyezésekor arra kell ügyelnünk, hogy a gyorstárakon kívül is módosítható adatokat soha ne keverjük össze a processzor által normál módon használtakkal. Ha nincs más lehetőség, mint ennek a szabálynak a felrúgása, akkor ügyeljünk arra, hogy az adatszerkezet minden külsőleg módosítható elemének elhelyezését az `L1_CACHE_BYTES` makró segítségével a gyorstár sorhosszához igazítsuk. Vizsgáljuk meg, hogy a kód futtatására kiszemelt processzortípusok közül melyik rendelkezik a legnagyobb gyorstársorméretttel és a makró értékét ezzel megegyezőre állítsuk. A legjobb, ha `kmalloc` segítségével két külön területet foglalunk le. A `kmalloc` soha nem foglal olyan területeket, amelyek azonos gyorstársorba lapolódnak át.

Gyorstárkezelő utasítások

A legalapvetőbb utasítás az `invalidate`, az érvénytelenítés, amely a megadott sort az összes gyorstárból törli. A parancs kiadását követően a törölt adatokra való hivatkozáskor a rendszer azokat újratölti a főmemóriából. Az adatok elavulásával kapcsolatos gondok tehát úgy háríthatók el, hogy a kérdéses gyorstársort az adatolvasás előtt érvénytelenítjük. Linux alatt az érvénytelenítés a következőképpen történik:

```
void
dma_cache_inv(unsigned long cím
              unsigned long méret);
```

A cím az a virtuális cím, amelyen a műveletet el kell kezdeni, a méret pedig az érvénytelenítendő adatrész hosszát adja meg. Megjegyzem, a méretet a gép önműködően a gyorstár-sorméret egész értékű többszörösére kerekíti fel.
A visszaírásos gyorstáraknál a főmemóriába tetszőleges piszkos sor kiírására, más szóval kiürítésére van módunk:

```
void
dma_cache_wback(unsigned long cím,
                unsigned long méret);
```

A kiürítést azelőtt kell elvégezni, hogy bármilyen változtatás történne a piszkos gyorstársorhoz tartozó főmemóriabeli részen. Másként szemlélve a feladatot: a kiürítést azt megelőzően kell végrehajtani, hogy bármilyen külső eszköz, például egy PCI kártya a főmemória tartalmát módosítaná; illetve a kiürítés után egy érvénytelenítési parancsot is ki kell adnunk, mielőtt a processzor valamilyen megváltozott adathoz hozzáférne.

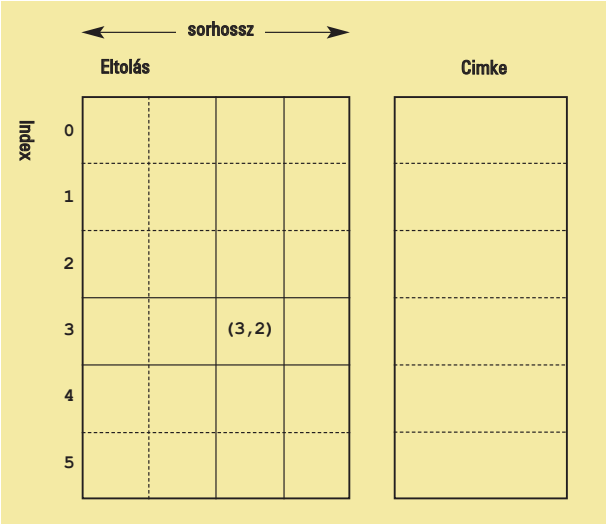
Elméletileg a visszaírásos gyorstáraknál az érvénytelenítés az adatokat a kiírásuk nélkül gyilkolja ki , amelyek így lényegében megsemmisülnek. Ezért biztonságosabb, ha kiürítő és egyben érvénytelenítő utasítást adunk ki:

```
void
dma_cache_wback_inv(unsigned long cím,
                    unsigned long méret);
```

A parancs hatására a rendszer a gyorstársorok tartalmát a főmemóriába másolja, majd a bennük lévő adatokat érvénytelennek nyilvánítja.

Gyorstártípusok

Ebben a részben a gyorstárak tényleges működéséről lesz szó. Mindebből az egyetlen nélkülözhetetlen adat az álnevesítés fogalma, amely azt a jelenséget takarja, amikor a memória adott fizikai címen elérhető tartománya több különböző gyorstársorban is szerepel. Arról, hogy a rendszermag hogyan kezeli az álneveket, a következő szakaszban lesz szó.
A közvetlenül leképezett gyorstáraknál – lásd a *3. ábrát* – a tár meghatározott hosszúságú sorokra bontható (a példában a sorhossz négy). A gyorstár mindegyik sora egyedi indexet kap, vagyis a gyorstárban található bájtok mindegyikét a sorindex és a soron belüli eltolás ismeretében tudjuk megcímezni. Minden indexhez egy rejtett érték is tartozik, ez a címke.
A rendszer minden címe három részre bontható: címke, index és eltolás (*4. ábra*). Amikor egy sor bekerül a gyorstárba, a gép a címből kiemeli a címkét és az indexet. A sor a megfelelő indexszel kerül elhelyezésre a gyorstárban, a címke tárolása pedig rejtetten, a sorban lévő adatokkal együtt történik. Amikor a processzor egy adott címre hivatkozik, a gép a megadott index alapján keres a gyorstárban. Ha a címkék egyeznek, akkor a soron belül az eltolás alapján történik a hivatkozás szerinti rész kikeresése. Ha a címkék nem egyeznek, akkor a meglévő sort ki kell üríteni a főmemóriába, a helyes sort pedig be kell tölteni a gyorstárba. Ilyenkor minden gyorstárazható címhez csak egyetlen indexsor tartozhat, ami sokszor hátrányos. Ha például a processzor olyan címsorozatról olvas, amelynek a tagjai véletlenül azonos gyorstárbeli indexszel rendelkeznek, a gyorstársort minden egyes olvasásnál ki kell üríteni és újra fel kell tölteni. Az ilyen helyzetek nem is olyan ritkák, gondoljunk például egy ciklusra, amely egy olyan adatszerkezet elemeit olvassa be, amelynek a mérete nagyjából a gyorstáréval egyezik. A közvetlenül leképezett gyorstárak indexét időnként gyorstárszínnek (cache-line coloring) is nevezik, a hibajelenséget pedig sorszínezési nehézségnek hívják.
A közvetlenül leképezett gyorstárak színezési nehézségének megoldására a gyorstárak áramköreit sokszor úgy alakítják ki, hogy a gyorstárban végzett keresésnél egyszerre egynél több sor címkéjét is meg lehessen vizsgálni. Egy N utas asszociatív gyorstárnál minden index N darab gyorstársorral – és címkével – mutat egyezést, vagyis egyszerre akár N darab azonos indexű cím is lehet a gyorstárban. A párhuzamos gyorstárbeli keresést lehetővé tévő áramkör természetesen növeli a gyártás költségét, ezért ilyet inkább a felsőbb kategóriás processzorokban találunk.
A csúcs a teljesen asszociatív gyorstár – az ilyen gyorstárakban egyáltalán nincs index, egy adott címke megkeresésekor az összehasonlítás az összes sorral egyszerre történik.



3. ábra Közvetlen leképezésű gyorstár

4. ábra Címfelosztás

A korszerű processzorok mindegyike címfordítást használ, amelynél a rendszermag vagy az alkalmazások által hivatkozott képzetes címek nem egyeznek meg azzal a fizikai címmel, amely alatt az adatok valójában megtalálhatók. A gyorstárat a címfordító egység elé és mögé is lehet helyezni, a többszintű gyorstáraknál gyakran vegyes megoldásokat találunk. Mindegyik megoldás más és más jellemzőkkel bír, más és más lehetőségeket kínál. Az alábbiakban ezeket tekintjük át.
A fizikai indexelésű, fizikai címkézésű (Physically Indexed, Physically Tagged – PIPT) gyorstáraknál a címke és az index egyaránt a fizikai memóriára utal. Ez a megoldás elegáns és egyszerű, csakhogy a PIPT gyorstárak alkalmazásánál érvényes címfordítási bejegyzésnek kell szerepelnie a processzor fordítási-előretekintő átmeneti tárában (Translation Lookaside Buffer, TLB). Ha a TLB bejegyzését a címfordítás elvégzése előtt kell kikeresni a memóriából, a gyorstár meglétéből fakadó előnyök lényegében elvesznek. Mégha van is megfelelő bejegyzés a TLB-ben, először benne kell keresni, majd utána a gyorstárban is, emiatt az ilyen megoldások lassúak.
A képzetesen indexelt, képzetesen címkézett (virtually indexed, virtually tagged – VIVT) gyorstárakban az index és a címke egyaránt képzetes-címtérbeli. Bár ilyenkor a gyorstárban lényegesen gyorsabban lehet keresni, hiszen ha a keresés sikeres, akkor sem előtte, sem utána nem kell címfordítást végezni, ez a megoldás is számos kérdést vet fel:

1. A képzetes címfordítások a rendszermag normál működése közben általában változnak, így a gyorstárnak figyelnie kell a TLB-bejegyzések, illetve a címtér változásait, és ezeknek megfelelően ki kell ürítenie azokat a sorokat, amelyeknek a fordítása megváltozott.
2. Mégha egyetlen címtér van is, ugyanahhoz a fizikai címhez több képzetes cím is tartozhat. A képzetes címek mindegyike külön is bekerülhet a gyorstárba, függet-

lenül attól, hogy ugyanarra az adatra hivatkoznak. Ezt nevezzük gyorstársor-álnevesítési nehézségnek (cache-line aliasing problem).

A VIVT gyorstárak nagyobb keresési sebességéből fakadó előnyök általában bőven ellensúlyozzák az ilyen memóriák használatából fakadó hátrányokat, így a nem x86 típusú processzoroknál rendszerint ilyen megoldásokkal találkozunk. Létezik egy hibrid gyorstártípus is, amely a VIVT gyorstárak hátrányait próbálja kiküszöbölni úgy, hogy lényeges sebességromlást ne okozzon – ezek a képzetesen indexelt, fizikailag címkézett (Virtually Indexed, Physically Tagged – VIPT) gyorstárak. Ezeknél az index képzetes cím, a címke viszont fizikai, így a címke–eltolás kettősnek kell a teljes fizikai címet megadnia. Emiatt a címkék más gyorstártípusoknál nagyobb méretűek.
A VIPT gyorstárak gyorsabbak, mint a PIPT típusúak, esetükben ugyanis a címfordítás és a gyorstárbeli keresés egyszerre is elvégezhető. Igaz, a processzor a címfordítás elvégzéséig nem tudja, hogy a gyorstársor érvényes-e, vagyis a címkék egyeznek-e.
A VIVT vonatkozású hátrányok eltűnnek, hiszen a címke fizikai, vagyis a VIPT gyorstár az álnevesítést önműködően felismeri, amikor azt látja, hogy a gyorstárban két azonos címke szerepel. A VIPT gyorstár tehát felépíthető úgy, hogy gyorstársor-álnevesítés soha ne történjen. Elméletileg létezik egy negyedik gyorstártípus is, ami fizikai indexelésű, képzetes címkézésű (Physically Indexed, Virtually Tagged – PIVT), ám ez lényegében semmire nem jó, így nem is foglalkozunk vele.

Az álnevesítési nehézség

Minden alkalommal, amikor a rendszermag a megadott fizikai laphoz egynél több képzetes leképezést hoz létre, gyorstársor-álnevesítés történik. Szerencsére a rendszermag az ilyen helyzeteket gondosan kerüli, így álnevesítés csak egyetlen esetben történik, amikor a felhasználó `mmap` segítségével használ egy fájlt. Ekkor a rendszermag egy képzetes címmel rendelkezik a fájlnak a lapgyorstárban lévő lapjaihoz, és a felhasználó is rendelkezik egy vagy több különböző képzetes címmel. Mindez azért lehetséges, mert a felhasználót semmi nem akadályozza meg abban, hogy a fájlt az `mmap` segítségével több helyen is elérhetővé tegye. Az `mmap` használatakor a rendszermag a fájlleíró listák egyikéhez adja hozzá a leképezést; ha az adatok módosítása nem megengedett, akkor az `i_mmap` listát, ha pedig az adatmódosítás is lehetséges, akkor az `i_mmap_shared` listát bővíti. Megadott lap gyorstárálneveit a következő hívással hozhatjuk összhangba:

```
void flush_dcache_page(struct page *page);
```

A hívást minden alkalommal meg kell ejteni, amikor egy lap tartalmát a rendszermag módosítja, és erre a hívásra a lap adatainak olvasása előtt kell sort keríteni, feltéve, hogy a `page->mapping->i_mmap_shared` nem üres. A géptípusoktól függő kódoknál a `flush_dcache_page` végighalad az `i_mmap_shared` listán és a gyorstárból kiüríti az adatokat. Ezt követően az `i_mmap` listán is végigmegy és érvényteleníti – ezzel minden álnevet egységesít.

Elkülönülő utasítás- és adatgyorstárak

A lehető legnagyobb hatékonyságra törekedve a processzorok gyakran külön gyorstárat alkalmaznak a végrehajtott utasítások és az általuk feldolgozott adatok tárolására. A két gyorstár sokszor teljesen elkülönül, és például az adatírásokat az utasításgyorstár nem is látja. Mindez akkor okoz gondot, ha olyan utasításokat próbálunk végrehajtani, amelyeket csak az előző pillanatban írtunk be a memóriába – ilyen például modul betöltésekor vagy ugródeszka (trampoline) használatakor fordul elő. A következő hívást kell használnunk:

```
void
flush_icache_range(unsigned long kezdet,
                   unsigned long vég);
```

A hívással biztosíthatjuk, hogy a végrehajtás megkezdése előtt az utasítások az utasításgyorstár számára is láthatóvá váljanak. A kezdet és a vég a módosított, az utasításokat tartalmazó memóriablokk kezdő és a záró címét adja meg.

A gyorstár általános kiürítése

A processzorok gyorstárainak általános kiürítésére két hívás használható:

```
void flush_cache_all(void);
```
és
```
void flush_cache_mm(struct mm_struct *mm);
```

Az első hívás a rendszer összes gyorstárát kiüríti, a második viszont csak azokat a sorokat, amelyek az `mm` mutatóval megadott folyamat címteréhez tartoznak. Mindkét hívás végrehajtása rendkívül költséges, ezért csak akkor alkalmazzuk őket, ha erre mindenképp szükség van.

Gyorstárazás SMP környezetben

Ha egynél több processzor van a rendszerben, akkor valamilyen szintig saját gyorstárral rendelkeznek.
A géptípustól függően lehetséges, hogy a rendszermag feladata annak biztosítása, hogy az egyik processzor által a saját gyorstárában végrehajtott változások a többi processzor számára is láthatók legyenek. Szerencsére a legtöbb processzor az ilyen egységességi kérdéseket hardveresen kezeli. Ha az adott processzor nem is ilyen, az itt ismertetett hívások használatakor nem kell attól tartanunk, hogy a processzorok közötti egységességgel gondjaink lennének.

Összegzés

Remélem, rövid összefoglalóm alapján mindenki át tudta tekinteni a gyorstárak működését és azt, hogy a rendszermag hogyan kezeli őket.

Linux Journal 2004. január, 117. szám

James Bottomley
A SteelEye programtervezője. A SCSI alrendszer, a Linux Voyager-átültetés és az 53c700 illesztőprogram karbantartásáért felel. DMA/eszközmodellezési-elvonatkoztatási munkájával a PA-RISC Linux fejlesztéséhez is hozzájárult.

I2C illesztőprogramok (2. rész)

Lássuk, hogyan tudjuk leolvasni a gép hőmérsékletét, a ventilátorok fordulatszámát és az egyéb fontos jellemzőket mérő érzékelők jeleit!

Sorozatunk első részében az I2C buszillesztőprogramok és az I2C algoritmus-illesztőprogram működését tárgyaltuk, valamint egy egyszerű I2C buszillesztőprogramot is összeállítottunk. Ebben a hónapban az I2C lapka-illesztőprogramok működését tekintjük át, majd üzem közben is megvizsgálunk egyet.

Az I2C lapka-illesztőprogramok az I2C-buszra csatlakozó I2C-eszközökkel folytatott párbeszédeket vezérlik. Az I2C-lapkák általában az alaplapra csatlakozó különféle rendszerelemek fizikai jellemzőit figyelik, például a ventilátorok fordulatszámát, a hőmérsékleti és a feszültségértékeket.

Az `i2c_driver` adatszerkezet egy I2C lapka-illesztőprogramot ír le. Megadása az ***include/linux/i2c.h*** fájlban található. Működő illesztőprogram létrehozásához mindössze a következő mezőkre van szükségünk:

- `struct module *owner;` – értéke `THIS_MODULE`, segítségével a modulra való hivatkozások számlálása oldható meg.
- `char name[I2C_NAME_SIZE];` – az I2C lapka-illesztőprogram beszédes neve. Ez az érték jelenik meg az egyes I2C-eszközökhöz létrehozott sysfs alatti ***name*** fájlokban.
- unsigned int flags; – értékét az `I2C_DF_NOTIFY` értékével megegyezően kell beállítani, így a lapka-illesztőprogram értesülhet arról, ha betöltése után további I2C-eszközök betöltésére is sor kerül. Ez a mező hamarosan valószínűleg eltűnik, mivel az illesztőprogramok túlnyomó része maga gondoskodik a beállításáról.
- `int (*attach_adapter)(struct i2c_adapter *);` – meghívására akkor kerül sor, amikor a rendszer új I2C buszillesztőprogramot tölt be. Később még bővebben is szó lesz róla.
- `int (*detach_client)(struct i2c_client *);` – meghívására az `i2c_client` eszköz rendszerből való eltávolításakor kerül sor. Kicsit később részletesebben is kitérünk rá.

Az alábbi kódrészlet a ***tiny_i2c_chip.c*** nevű I2C példalapka-illesztőprogramból származik. A program a Linuxvilág honlapjáról (➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/I2C könyvtára) címről tölthető le. Az `i2c_driver` adatszerkezet a következőképpen épül fel:

```
static struct i2c_driver chip_driver = {
    .owner          = THIS_MODULE,
    .name           = "tiny_chip",
    .flags          = I2C_DF_NOTIFY,
    .attach_adapter = chip_attach_adapter,
    .detach_client  = chip_detach_client,
};
```

Az illesztőprogram bejegyzése

Az I2C lapka-illesztőprogram bejegyzéséhez az `i2c_add_driver` függvényt kell meghívni, átadott értéke egy mutató lesz az `i2c_driver` adatszerkezetre:

```
static int __init tiny_init(void)
{
    return i2c_add_driver(&chip_driver);
}
```

Az I2C lapkaillesztőprogram bejegyzésének törlését az `i2c_del_driver` függvénnyel végezhetjük el, ennek átadott értéke szintén az `i2c_driver` adatszerkezetre mutat:

```
static void __exit tiny_exit(void)
{
    i2c_del_driver(&chip_driver);
}
```

Az I2C lapka-illesztőprogram bejegyzését követően az I2C buszillesztőprogram betöltésekor az `attach_adapter` visszahívó (callback) függvény meghívására kerül sor. Ez a függvény ellenőrzi, hogy látható-e valamilyen I2C-eszköz azon az I2C-buszon, amelyhez az ügyfél-illesztőprogram csatlakozni próbál. A vizsgálathoz szinte minden I2C lapka-illesztőprogram az I2C-mag `i2c_detect` függvényét használja. A ***tiny_i2c_chip.c*** illesztőprogram például a következőképpen jár el:

```
static int
chip_attach_adapter(struct i2c_adapter *adapter)
{
    return i2c_detect(adapter,
           &addr_data,chip_detect);
}
```

Az `i2c_detect` függvény az `addr_data` adatszerkezetben megadott címeken végez próbákat az I2C-csatolóval. Ha talál

valamilyen eszközt, a `chip_detect` függvényt hívja meg. Ha közelebbről is megvizsgáljuk a forráskódot, észrevehetjük, hogy az `addr_data` adatszerkezet sehol nem szerepel. Ennek az az oka, hogy az adatszerkezet létrehozása a `SENSORS_INSMOD_1` makró feladata. A makró megadása az ***include/linux/i2c-sensor.h*** fájlban található, és nem nevezném könnyed délutáni olvasmánynak. Az illesztőprogram által támogatott különféle lapkatípusok és az elérésükre alkalmas címek alapján a makró egy `addr_data` nevű statikus változót hoz létre. Ezeket az értékeket később a modulnak átadott értékek segítségével lehet megváltoztatni. Egy I2C lapka-illesztőprogramnak a következő változókat kell rendelkezésre bocsátania: `normal_i2c`, `normal_i2c_range`, `normal_isa` és `normal_isa_range`. Ezek a változók adják meg az illesztőprogram által támogatott i2c smbus és i2c isa címeket. Valójában címeket tartalmazó tömbökről van szó – mindegyiket egy különleges érték zárja, az `I2C_CLIENT_END` és az `I2C_CLIENT_ISA_END`. Egy-egy I2C lapkatípus általában csak korlátozott számú címen bukkanhat fel.
A ***tiny_i2c_client.c*** illesztőprogram az említett változókat a következő módon adja meg:

```
static unsigned short normal_i2c[] =
  { I2C_CLIENT_END };
static unsigned short normal_i2c_range[] =
  { 0x00, 0xff, I2C_CLIENT_END };
static unsigned int normal_isa[] =
  { I2C_CLIENT_ISA_END };
static unsigned int normal_isa_range[] =
  { I2C_CLIENT_ISA_END };
```

A `normal_i2c_range` változó azt tudatja velünk, hogy ezt a lapkát bármelyik I2C smbus címen megtalálhatjuk, vagyis illesztőprogramunkat gyakorlatilag bármelyik I2C buszillesztőprogrammal ki tudjuk próbálni.

Mit kell tenni a lapka megtalálása után?

A ***tiny_i2c_chip.c*** illesztőprogramnál az I2C-eszköz megtalálása után az I2C-mag a `chip_detect` függvényt hívja meg. Ennek megadásában a következő átadott értékek szerepelnek:

```
static int
chip_detect(struct i2c_adapter *adapter,
            int address, int kind);*
```

Az `adapter` változó azt az I2C adapter-adatszerkezetet adja meg, amelyen a lapka fellelhető. Az `address` változó azokat a címeket tartalmazza, amelyeken a lapka elérhető, a `kind` változó pedig a megtalált lapka pontos típusát jelzi. A `kind` változót a legtöbb esetben figyelmen kívül hagyja a rendszer, de vannak olyan I2C lapka-illesztőprogramok, amelyek különböző típusú I2C-lapkákat is támogatnak, ezek pontosan a `kind` segítségével tudakolják meg a gépben található lapka típusát.
Ez a függvény felelős az `i2c_client` adatszerkezet létrehozásáért, amely később az I2C-magnál bejegyzésre kerül. Az I2C-mag az adatszerkezetet önálló I2C-eszközként használja. Az adatszerkezet létrehozását a `chip_detect` függvény az 1. listán látható módon végzi el (➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/I2C).

Első lépésként az `i2c_client` adatszerkezet és `chip_data` névvel egy különálló, helyi adatszerkezet jön létre és kap nulla kezdőértéket. Fontos, hogy az `i2c_client` kezdeti nullás értékadása megtörténjen, ugyanis ennek hiányában előfordulhat, hogy a rendszermag-illesztőprogram alsó rétegei nem működnek helyesen. A memória sikeres lefoglalását követően az `i2c_client` adatszerkezet néhány mezőjének az értékét úgy módosítjuk, hogy a tényleges eszközre és illesztőprogramra mutassanak. Kicsit kézzelfoghatóbban: az `addr`, az `adapter` és a `driver` változók kezdeti értékadását kell elvégezni. Az `i2c_client` adatszerkezet nevét úgyszintén be kell állítani, feltéve, hogy azt szeretnénk, hogy a `sysfs` fának az I2C-eszközhöz tartozó bejegyzése helyesen jelenjen meg.
Az `i2c_client` adatszerkezetet kezdeti értékadása után be kell jegyezni az I2C-magnál, ezt az `i2c_attach_client` függvénnyel tehetjük meg:

```
/* Szólunk az I2C-rétegnek, hogy új ügyfél
érkezett. */
err = i2c_attach_client(new_client);
if (err)
    goto error;
```

Ha a függvényhívás hibajelzés nélkül tér vissza, akkor az I2C-eszköz üzembe helyezése a rendszermag oldaláról megtörtént.

I2C és sysfs

A 2.0-s, 2.2-es és 2.4-es sorozatszámot viselő rendszermagokban az I2C-kód az I2C-eszközöket a ***/proc/bus/i2c*** könyvtárba helyezi. A 2.6-os rendszermagnál minden I2C-eszköz és -csatoló a `sysfs` fájlrendszerben jelenik meg. Az I2C-eszközök a ***/sys/bus/i2c/devices*** elérési úton találhatók meg, csatoló- és lapkacímek szerint rendezve.
Ha például egy gépen betöltjük a `tiny_i2c_chip` illesztőprogramot, ennek hatására a 2. listán látható (➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/I2C) `sysfs` faszerkezet jön létre.

Négy különböző I2C-eszköz jelenik tehát meg, s mindegyiket ugyanaz a `tiny_chip` illesztőprogram kezeli. A vezérlést végző illesztőprogramot úgy találhatjuk meg, hogy átnézzük a ***/sys/bus/i2c/drivers*** könyvtár tartalmát vagy belépünk a lapka eszköz saját könyvtárába, és megnézzük a ***name*** fájl tartalmát:

```
$ cat /sys/devices/pci0000:00/0000:00:06.0/
i2c-0/0-0009/name
tiny_chip
```

Az I2C lapka-illesztőprogramok a különféle érzékelőkről leolvasott értékeket az I2C-eszköz könyvtárában lévő `sysfs` fájlokon keresztül teszik elérhetővé. A fájlok neve szabványosított, ahogy az értékek értelmezésére használt mértékegységek is. A vonatkozó leírások a rendszermagfa ***Documentation/i2c/sysfs-interface*** állományában találhatók (lásd a *táblázatot*).
Mint az 1. táblázat alapján is látható, minden fájlban csak egy érték szerepel. Az összes fájl olvasható, továbbá bizo-

Az érzékelők sysfs fájlokon keresztül elérhető értékei

temp_max[1-3]	A legmagasabb hőmérséklet fixpontos, XXXXX formátumú érték, ezerrel elosztva Celsius-fokban mért hőmérsékletet kapunk. Olvasható és írható érték.
temp_min[1-3]	A legmagasabb hőmérséklet- vagy hiszterézisérték fixpontos, XXXXX formátumú érték, ezerrel elosztva Celsius-fokban mért hőmérsékletet kapunk. Általában hiszterézisérték, abszolút hőmérsékletet jelent, és nem a legnagyobbtól való eltérést ad meg. Olvasható és írható érték.
temp_input[1-3]	Bemeneti hőmérsékleti érték. Csak olvasható.

nyos fájlokat – a megfelelő jogosultság birtokában – a felhasználók írhatnak is.
A ***tiny_i2c_chip.c*** illesztőprogram egy I2C-eszközt emulál, amely képes a hőmérsékleti értékek jelentésére. A `sysfs` alatt létrehozza a ***temp_max1***, a ***temp_min1*** és a ***temp_input1*** fájlt; az általa visszaadott érték a fájlok minden egyes olvasásakor növekszik, így az egyes lapkaértékek könnyen megkülönböztethetők egymástól.
A `sysfs` alatt fájlt létrehozni a `DEVICE_ATTR` makróval lehet:

```
static DEVICE_ATTR(temp_max, S_IWUSR | S_IRUGO,
                   show_temp_max, set_temp_max);
static DEVICE_ATTR(temp_min, S_IWUSR | S_IRUGO,
                   show_temp_hyst, set_temp_hyst);
static DEVICE_ATTR(temp_input, S_IRUGO,
                   show_temp_input, NULL);
```

A makró egy adatszerkezetet hoz létre, amely a `chip_detect` függvény futásának végén a `device_create_file` függvénynek kerül átadásra:

```
/* sysfs fájlok bejegyzése */
device_create_file(&new_client->dev,
                   &dev_attr_temp_max);
device_create_file(&new_client->dev,
                   &dev_attr_temp_min);
device_create_file(&new_client->dev,
                   &dev_attr_temp_input);
```

Ezzel a hívással létrejönnek az eszközhöz tartozó `sysfs` fájlok:

```
/sys/devices/pci0000:00/0000:00:06.0/i2c-0/0-0009
|-- detach_state
|-- name
|-- power
|   `-- state
|-- temp_input
|-- temp_max
`-- temp_min
```

A ***name*** fájlt az I2C-mag hozza létre, a `detach_state` és a ***power/state*** fájlokat pedig az illesztőprogrammag.
Térjünk azonban vissza egy kicsit a `DEVICE_ATTR` makróra! A makrónak ismernie kell a létrehozandó fájl nevét, a rá

3. lista Függvényt létrehozó makró

```
#define show(value)
static ssize_t
show_##value(struct device *dev, char *buf)
{
  struct i2c_client *client = to_i2c_client(dev);
  struct chip_data *data =
      i2c_get_clientdata(client);

  chip_update_client(client);
  return sprintf(buf, "%d\n", data->value);
}
show(temp_max);
show(temp_hyst);
show(temp_input);
```

4. lista A set_temp_max függvény

```
#define set(value, reg)
static ssize_t
set_##value(struct device *dev,
            const char *buf, size_t count)
{
  struct i2c_client *client = to_i2c_client(dev);
  struct chip_data *data =
      i2c_get_clientdata(client);
  int temp = simple_strtoul(buf, NULL, 10);

  down(&data->update_lock);
  data->value = temp;
  up(&data->update_lock);
  return count;
}
set(temp_max, REG_TEMP_OS);
set(temp_hyst, REG_TEMP_HYST);
```

vonatkozó futtatási, olvasási és elérési engedélyeket, valamint annak a függvénynek a nevét, amelyet a fájl olvasásakor és írásakor meg kell hívni. A ***temp_max*** fájl esetében ezeknek a megadása a következőképpen történt:

```
static DEVICE_ATTR(temp_max, S_IWUSR | S_IRUGO,
                   show_temp_max, set_temp_max);
```

A fájl olvasásakor a `show_temp_max` függvényt kell meghívni; ennek megadása – sok más `sysfs` fájlhoz hasonlóan – egy másik, egy függvényt létrehozó makróval lehetséges (lásd a *3. listán*).
Annak oka, hogy a függvény létrehozása makróval történik, az, hogy így egyszerűen, a programkód többszörözése nélkül lehet további, gyakorlatilag ugyanolyan feladatokat ellátó, ám más névvel ellátott és más változók értékét kiolvasó sysfs fájlokat létrehozni. A makró három különböző függvényt hoz létre, ezekkel a `chip_data` adatszerkezet három különböző változójának az értékét lehet kiolvasni.
A függvény futása során a `device *` adatszerkezet

5. lista A detatch_client függvény

```
static int
chip_detach_client(
    struct i2c_client *client)
{
    struct chip_data *data =
       i2c_get_clientdata(client);
    int err;

    err = i2c_detach_client(client);
    if (err) {
      dev_err(&client->dev,
              "Ügyfél bejegyzésének törlése"
              " sikertelen, az ügyfél"
              " nem vált le.\n");
      return err;
    }
    kfree(client);
    kfree(data);
    return 0;
}
```

`i2c_client` * adatszerkezetté alakul át. Ezután a privát `chip_data` * adatszerkezet átvételére kerül sor az `i2c_client` * adatszerkezetből. A következő lépés a lapkaadatok frissítése a `chip_update_client` meghívásával. Onnan a kért változó egy átmeneti tárba (buffer) íródik, majd átadásra kerül az illesztőprogrammagnak, amely a felhasználónak adja tovább:

```
$ cat /sys/devices/.../0-0009/temp_input
1
```

A `chip_update_client` minden egyes meghívásakor eggyel növeli az összes értéket:

```
static void
chip_update_client(struct i2c_client *client)
{
    struct chip_data *data =
        i2c_get_clientdata(client);

    down(&data->update_lock);
    dev_dbg(&client->dev, "%s\n", __FUNCTION__);
    ++data->temp_input;
    ++data->temp_max;
    ++data->temp_hyst;
    data->last_updated = jiffies;
    data->valid = 1;
    up(&data->update_lock);
}
```

Így az értékre vonatkozó kérések mindegyike más lesz:

```
$ cat /sys/devices/.../0-0009/temp_input
2
$ cat /sys/devices/.../0-0009/temp_input
3
```

A `set_temp_max` függvény, amely a változók írását teszi lehetővé, szintén makró segítségével jön létre (lásd a *4. listán*).
A `show` függvényekhez hasonlóan ez a függvény is a `device` * adatszerkezetet `i2c_client` * adatszerkezetté alakítja, ezt követően kerül sor a privát `chip_data` * adatszerkezet létrehozására. Ezután a felhasználó által átadott adatokból a `simple_strtoul` függvény segítségével egy számot állítunk elő, majd a megfelelő változóba mentjük:

```
$ cat /sys/devices/.../0-0009/temp_max
1
$ echo 41 > /sys/devices/.../temp_max
$ cat /sys/devices/.../0-0009/temp_max
42
```

Rendrakás

Az I2C lapka-illesztőprogram rendszerből való eltávolításakor, történjen az akár az I2C buszillesztőprogram, akár az I2C lapka-illesztőprogram eltávolításával, az I2C-mag meghívja az `i2c_driver` adatszerkezetben megadott `detatch_client` függvényt. Ez – mint a példa-illesztőprogramból is kitűnik (*5. lista*) – általában egy egyszerű függvény.
Ahogy az `i2c_attach_client` függvény hívására volt szükség az `i2c_client` adatszerkezetnek az I2C-magnál történő bejegyzéséhez, úgy az `i2c_detach_client` függvény végzi el a bejegyzés törlését. Ha a függvény végrehajtása sikerrel jár, akkor – még a függvényből való visszatérés előtt – fel kell szabadítani az illesztőprogram által az I2C-eszköz számára lefoglalt memóriát.
A példa-illesztőprogram nem végzi el a `sysfs` fájlok kifejezett eltávolítását a `sysfs`-magból, erre az illesztőprogrammagban önműködően kerül sor, az `i2c_detach_client` függvény részeként. Ha a szerző úgy kívánja, a `device_remove_file` meghívásával a fájlok eltávolítását kézzel is elvégezheti.

Összegzés

Két írásomban áttekintettem a rendszermag I2C illesztőprogramok írásával, valamint az I2C algoritmus- és az I2C lapka-illesztőprogrammal kapcsolatos alapvető tudnivalókat. Az I2C illesztőprogramok írásával kapcsolatban rengeteg érdekes olvasnivaló található a rendszermagfa ***Documentation/i2c*** könyvtárában és az ***Lm_sensors*** oldalon (➲ http://secure.netroedge.com/~lm78).

A cikkhez kapcsolódó listák megtalálhatóak a ➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/I2C könyvtárában.

Linux Journal 2004. február, 118. szám

Greg Kroah-Hartman (greg@kroah.com)
Jelenleg a Linux-rendszermag különféle illesztőprogram-alrendszereiért felelős. Az IBM-nél dolgozik és a Linux-rendszermaggal kapcsolatos kérdésekkel foglalkozik.

HA-OSCAR: a magas rendelkezésre állású OSCAR

Ha egy géptelep mérete eléri a több ezer csomópontot, meg kell szüntetni a leggyengébb láncszemeket. A HA-OSCAR a megoldás.

Az Open Cluster Group (OCG) és a belőle kifejlődött Open Cluster Application Resources (OSCAR) projekt magját néhány hasonlóan gondolkodó ember ültette el anno 2000. február 17-én, a Department of Energy által támogatott tanácskozást követő vacsorán.
Ez a csoport vitatta meg fehér asztal mellett, hogy milyen programok szükségesek egy nagy teljesítményű Beowulf-géptelep felépítéséhez. A csoport egyetértett abban, hogy egyszerű feladat géptelepet úgy létrehozni, hogy vezetékekkel sok közönséges számítógépet kötnek össze. Megállapították azt is, hogy a Beowulf-géptelep telepítése, s a programkörnyezet beállítása túlságosan nagy erőfeszítést igényel. Ez sokaknak csak nagy nehézségek árán megy, egyszersmind mindenki számára fárasztó. Megszületett a folyamat egyszerűsítésének ötlete.
Az elképzelés tovább érlelődött Oak Ridge-ben (Tennessee), ugyanazon év áprilisában egy másik tanácskozáson, ahol az ipar, az akadémiai szféra és a kutatóintézetek képviselői vettek részt. Ez volt az első hivatalos alkalom, ahol az OCG-t megalakították, és elkezdtek dolgozni az OSCAR-on, különösen a Beowulf-géptelep programjain és a hozzá kapcsolódó telepítési folyamaton. Ezen a konferencián a csoport három alapelvben egyezett meg:

1. A géptelepek addig nem tudnak betörni a nagyteljesítményű számítások piacára, amíg nincsen olyan széles körben elfogadott és megbízható programkörnyezet, amit az átlagfelhasználó is képes kezelni.
2. Az OCG a nyílt forráskód modelljét használja, ezért az OSCAR csak szabadon terjeszthető kódot tartalmazhat, és előnyt élveznek a Berkeley-stílusú felhasználási szerződéssel terjesztett kódok.
3. Az OCG elérheti a célját, ha a jelenleg hozzáférhető legjobb gyakorlatot tükröző kódokat használja fel.

Az OCG és az OSCAR kezdetéről részletesebben is volt már szó a Linuxvilág hasábjain (2002. július – Az OSCAR-forradalom). Az OSCAR rövid története során a csoport mindvégig ki tudott tartani e három elv mellett, miközben másféle programokat is befogadtak az OSCAR-ba. Bár az OSCAR maga csak szabadon terjeszthető kódot tartalmaz, mások készíthetnek olyan OSCAR-csomagokat, amelyek nem részei ugyan az OSCAR-terjesztésnek, de egy meglévő OSCAR-telepítés alá felpakolhatók.

Az OSCAR projekten továbbra is vegyesen dolgoznak az ipar és a tudomány képviselői. A projekt vezetése a résztvevő szervezetek munkatársaiból kétévente megválasztott irányító testület kezében van.
Tavaly két OCG-munkacsoport is létrejött más géptelepkörnyezetek kutatására. Az új csoportok a saját géptelepváltozataikhoz felhasználják az OSCAR eredményeit; ez a két csoport a Thin-OSCAR és a HA-OSCAR. A Thin-OSCAR vezető tagja a kanadai Université de Sherbrooke – céljuk a merevlemez nélküli OSCAR kifejlesztése. A HA-OSCAR csoportot e cikk szerzői vezetik, és érdeklődésük homlokterében a magas rendelkezésre állású OSCAR létrehozása áll.

HA-OSCAR: küldetés, célok és emberek

2001 júliusában egy értekezletet tartottak a kanadai Ericsson Researchnél, ***Ibrahim Haddad*** ott vetette fel a magas rendelkezésre állás ötletét a géptelepekkel végzett számításokban. Az eredeti téma a magas rendelkezésre állás a távközlésben volt. Ahogy a beszélgetés haladt előre, világossá vált, hogy a bevezetni kívánt, több tízezer csomópontból álló, nagyteljesítményű számításokra használt géptelepeknél a magas rendelkezésre állás világából származó megoldások emelhetik a hibatűrés szintjét.
Ibrahim csoportja az Ericsson Researchnél lényegében egyedül dolgozott a magas rendelkezésre állás elérésén, amíg ***Dr. Chokchai Leangsuksun*** és csapata a Louisianai Műszaki Egyetemről nem csatlakozott hozzájuk, illetve ***Stephen Scott*** az ORNL-től el nem kezdett érdeklődni a HA-OSCAR iránt, még 2002 szeptemberében. 2002-ben a HA-OSCAR munkacsoportot hivatalosan is elismerte az OCG. A csoport elsődleges célja, hogy a meglévő OSCAR-megoldások felhasználásával a magas rendelkezésre állás tulajdonságával ruházzák fel az OSCAR géptelepeket.
A megoldásnak elsősorban a távközlésben és a nagyteljesítményű számítások világában lehetnek felhasználói.

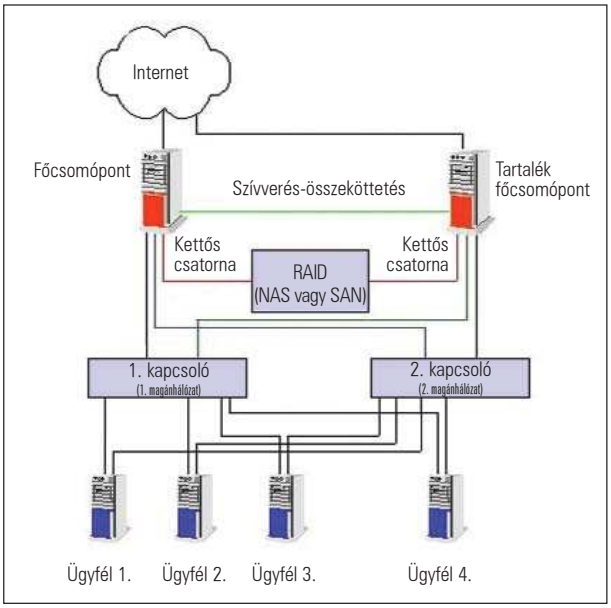
A HA-OSCAR többletszolgáltatásai az OSCAR-hoz képest

A HA-OSCAR az OSCAR-hoz képest számos továbbfejlesztést és új képességeket tartalmaz, főként a rendelkezésre állás, a méretezhetőség és a biztonság területén. A legtöbb új képesség megfeleltethető az ITU (International Telecommunication Union), a TMN (Telecommunication Management Network) és a FCAPS (Fault-management, Configu-

ration, Accounting, Performance and Security) ajánlásnak. Ezeket az ajánlásokat széles körben használják a távközlésben a hálózati eszközök kezelésére.

Két főcsomópont és tartalékolás

A szokásos géptelepeknél sok csomópont van, amelyek bizonyos fokú rendelkezésre állást biztosítanak; a legtöbbször viszont csak egyetlen főcsomópont van, amely nincs tükrözve, ezért könnyen lehet a leggyengébb láncszem. Az OSCAR jelenlegi kiadása is ilyen, és ez a létfontosságú rendszerekhez nem felel meg, mert számos olyan elem található benne, amely egyedi és hiba esetére nincs tartalék. A magas rendelkezésre állás (HA) követelményeinek kielégítéséhez a géptelepeket olyan képességekkel kell felruházni, amelyek megszüntetik a leggyengébb láncszemeket. Az eszközök többszörözése, a tartalék hálózati kapcsolat



A HA-OSCAR géptelepítő rendszer felépítése

mindennapos módszer a számítógéprendszerek megbízhatóságának és rendelkezésre állásának növelésére. A HA-OSCAR géptelepnél először a főcsomópont többszörözését szükséges megoldanunk. Ilyen elrendezés többféle módon is megvalósítható, például az aktív-aktív, aktív-meleg és aktív-hideg tartalék módszerével.

Az aktív-aktív módszer a teljesítményt és a rendelkezésre állást is növeli, mert mindkét főcsomópont egyszerre működik. A gond csak az, hogy ezt nehéz megvalósítani, és hiba esetén ellentmondásos adatok keletkezhetnek. Emiatt a másik két módszert terjedt el. Egy tartalék kiszolgáló figyeli az elsődleges kiszolgáló egészségi állapotát, és hiba esetén átveszi az irányítást. Jelenleg az aktív-meleg módszert használjuk. *Ábránk* a HA-OSCAR géptelepítő rendszer felépítését mutatja.

Kísérleteztünk Linux virtuális kiszolgálóval (LVS) és a Heartbeat- (szívverés) megoldással, és ezt terveztük beépíteni kiindulási aktív tartalék HA-OSCAR-változatunkba. Ezt a kiindulási felépítést a tartalékváltozat kiadása után kibővítenénk, hogy támogassa az aktív-aktív HA-t is.

Az aktív-aktív felépítés jobban kihasználja az erőforrásokat, mert mindkét főcsomópont egyszerre képes szolgáltatások nyújtására. A két főcsomópont hibatűrő DHCP, NTP, TFTP, NFS és SNMP-kiszolgálókat futtathat. Ha az egyik főcsomópont kiesik, minden általa nyújtott szolgáltatást a másik, tartalék főcsomópont vesz át, és csökkent teljesítménnyel szolgáltat tovább (elméletileg csúcsterhelésen ötven százalékos teljesítménnyel).

A HA-OSCAR másik HA-tulajdonsága a magas rendelkezésre állású hálózati összeköttetés, amit egyetlen gépbe épített több ethernetkapu használatával érünk el. Ráadásul a teljes hálózatot az összekötő elemeket (kapcsolók, vezetékek) is megkétszereztük. Ez teszi lehetővé, hogy a géptelep mindegyik csomópontja két vagy több úton is kapcsolódjon a hálózatához. A tartalék ethernetelemeknek köszönhetően a géptelep hálózati rendelkezésre állása nő; ráadásul ha mindkét hálózat működik, az adatcsere is növekszik, mert a redundáns kapcsolattartási útvonalakon például kihasználható az üzenetek csatornához kötésének módszere. A HA-OSCAR felhasználja más megvalósítások és már létező projektek eredményeit, például ilyen a High-Availability Linux, a Kimberlite és a Linux Virtual Server. A továbbfejlesztéseket és eredményeket vissza fogjuk juttatni a közösségnek.

Mindkét IP-verem támogatása (IPv4 és IPv6)

Az IPv6 az internetprotokollnak az IETF által tervezett következő nemzedéke, amely le fogja váltani a jelenleg használatos IPv4-et. Az interneten manapság többnyire az IPv4-et használják, ami korát meghazudtolóan rugalmas, de vannak vele gondok. A legfontosabb ilyen az IPv4-címek egyre növekvő hiánya, márpedig minden új, internetre kötött eszköznek címre van szüksége. Az IETF azért dolgozta ki az IPv6 szabványt, hogy az IPv4 gondjait megoldja és tovább is fejlessze azt a jövő internete számára. Ezek a továbbfejlesztések sok területen jelentkeznek, ilyen például az útválasztás, az önműködő beállítás, a biztonság, a QoS és a mobilitás.

A HA-OSCAR alapból támogatja az IPv6-ot. A legtöbb internetszolgáltató és távközlési társaság az IPv4 és az IPv6 közös használatával kísérletezik. Minden HA-OSCAR-géptelep-csomópont támogatja az IPv6-ot, és közvetlenül a hálózati segédprogramokba is alapvető IPv6-képességek vannak belefordítva.

A sérült lemezek kezelése

Az OSCAR feltételezi, hogy az ügyfélcsomópontok merevlemezei, amelyekre települ, hibátlanok. Ez nem mindig van így: egyes csomópontok merevlemezei hibásak lehetnek.

A HA-OSCAR ezzel is törődik, és nem feltételezi, hogy minden csomópont merevlemeze jó alap a telepítéshez. Emiatt a telepítés során különleges parancsfájlok futnak és a rendszermag programból megvalósított RAID-et használ. Ezzel párhuzamosan kifejlesztettük a merevlemez tartalmának egyeztetésére szolgáló parancsfájlokat is, így ha a merevlemez elromlik, az adat nem vész el. Ráadásul a telepítővarázsló megpróbálja kijavítani a sérült lemezt. A HA-OSCAR támogatja még az egyidejű működést, a merevlemez eltávolítását, illetve hozzáadását, valamint a programból megvalósított RAID-et. A programból megvalósított RAID engedélyezésével a HA-OSCAR-t futtató géptelepek növekedett adatbiztonsággal és jobb teljesítménnyel üzemelnek.

Linux virtuális kiszolgáló (LVS) és a Heartbeat beépítése

Már áttekintettük a HA-OSCAR vasának a felépítését. A legfontosabb újítás a kettős főcsomópont bevezetése, amely, ha az elsődleges főcsomópont kiesik, tartalék főcsomópontot ad. A tartalék vas azonban nem elég önmagában a HA-hoz, észlelési és helyreállítási képességekkel is fel kell szerelni. Létezik néhány megoldás, amely a kiesést észlelni tudja. Kipróbáltuk és kiválasztottuk a hibatűrő LVS-t. A megoldás részei: LVS, Linux Director Daemon (`ldirectord`), Heartbeat és Coda. A Linux virtuális kiszolgáló egy olyan program, amely a hálózati kapcsolatokat több kiszolgálóhoz továbbítja, így osztva el közöttük a terhelést. A Heartbeat az elsődleges főcsomópont kiesését figyeli a soros vonalon és UDP-kapcsolaton keresztül. A Coda egy hibatűrő osztott fájlrendszer. Ez a megoldás nemcsak magas rendelkezésre állást kínál, hanem terheléskiegyenlítést is. A további LVS-szolgáltatásokat viszont fejleszteni kell a HA-OSCAR-ban, így például a SIP, PBS és webszolgáltásokat. Egy külső „szívverésfigyelőt" szintén beépítettünk a hibakezelő rendszerbe, ami akkor riaszt (értesíti a rendszergazdát), ha minden leáll, például mindkét főcsomópont kiesik.

A géptelep biztonsága

Az OSCAR-t jelenleg főként magánhálózatokra telepítik, ahol a biztonsági kockázat alacsony. Ez annak köszönhető, hogy ezek a géptelepek a laboratórium falain kívüli hálózathoz nem csatlakoznak. Ha viszont a HA-OSCAR-t internetre kötött géptelepre telepítik, a biztonság létfontosságú. A biztonság nemcsak azért életbevágó az OSCAR és a HA-OSCAR számára, mert egy betörő elérheti a géptelepet és a rajta elhelyezett adatokat, hanem mert egy rossz szándékú betörő tönkreteheti a rendszert és annak rendelkezésre állását. Sok biztonsági megoldás létezi, a külső megoldásoktól (tűzfal) a belső megoldásokig (épségellenőrző program). Sajnos ezek mindegyike csak egyetlen csomópont esetében használható, és a géptelepet nem képes egyetlen egységként kezelni. A sokféle biztonsági megoldás telepítése, foltozása, beépítése és kezelése rengeteg időt vesz igénybe. A megemelkedett karbantartási nehézségek hamarosan a biztonság csökkenését vonják maguk után, ahogy a különböző elemek frissítései növelik az együttműködési gondokat. Következésképpen kezdeményeztük az Osztott biztonsági háttér (Distributed Security Infrastructure, DSI) nevű nyílt forrású projekt elindítását, hogy a távközlési szintű telepbe szervezett kiszolgálók számára megfelelő biztonsági megoldást szolgáltassunk. A DSI egy olyan biztonsági keretrendszer, amely a géptelepeken futó programok számára osztott biztonsági szolgáltatásokat nyújt, például hozzáférés-kezelést, valamint az adatcsere épségét és megbízhatóságát. Ellenőrző szolgáltatásokat is kínál az egyes folyamatok szintjéig részletezve. A HA-OSCAR sokkal sikeresebb lehet a távközlésben és más létfontosságú területeken, ha támogatja a fejlett biztonsági szolgáltatásokat; emiatt a HA-OSCAR a DSI-t átvette az Ericssontól.

Csomópontok dinamikus hozzáadása és eltávolítása

A HA-OSCAR támogatja a géptelep csomópontjainak menet közbeni hozzáadását és eltávolítását. Ezek a műveletek a felhasználó számára észrevehetetlenek, a futó alkalmazásokat sem zavarják. Két nyílt forrású projekt nyújt hasonló szolgáltatást: az Eddie, amely az Ericsson nyílt forrású kezdeményezése, valamint az LVS. Jelenleg azt vizsgáljuk, hogy melyik a legjobb megoldás, és be fogjuk építeni azt a HA-OSCAR-ba. A célunk az, hogy észrevétlen művelet legyen, ha a nagyobb terhelés elviselésére új csomópontokat adunk hozzá, illetve a karbantartás idejére csomópontokat távolítunk el, és ne legyen hatással a rendszer rendelkezésre állására.

Linux-rendszermag

Felmerült a kérdés, hogy melyik rendszermagot használjuk, és hogy magunk foltozzuk-e a HA-OSCAR rendszermagját, vagy próbáljuk meg bejuttatni a foltjainkat a fő rendszermagfába. Úgy döntöttünk, hogy a legújabb megbízható 2.4-es rendszermagot fogjuk használni, és az általunk készített foltokat beküldjük a rendszermag levelezőlistájára. A HA-OSCAR-rendszermag építését egy segédprogram segítségével megpróbáljuk leegyszerűsíteni. A felhasználók a rendszermagot maguknak a helyi beállításaiknak megfelelően fordíthatják újra.

Hálózati fájlrendszerek támogatása

A géptelep építésének egyik alapköve a hálózati, illetve osztott fájlrendszer. Sok nyílt forrású projekt célja a linuxos géptelepek hálózati fájlrendszerének megvalósítása. A korábbi kutatásainkból és kísérletezéseinkből azt a következtetést vontuk le, hogy a géptelepen futó alkalmazástól függően más-más fájlrendszer felel meg a célnak. Például a párhuzamos virtuális fájlrendszer (PVFS) a nagy állományok mozgatásában jeleskedik, ezért hang- vagy videokiszolgálóhoz kiváló. Másfelől a beállításfájlok megosztása a géptelep csomópontjai között NFS-sel is elérhető, ehhez nem kell nagy adatátviteli teljesítmény. Ha a magas rendelkezésre állás és a SAN-ok támogatása fontos, akkor a naplózási képességekkel felruházott OpenGFS a megfelelő. Emiatt a HA-OSCAR minden lehetséges hálózati fájlrendszert támogatni fog, amelyet a célkörnyezetben használni lehet.

Gyors géptelep-telepítés

Fontos tényező az az idő, amelyet a géptelep megépítésére, elindítására és üzembe helyezésére kell fordítani. Ez kis géptelepeknél nem nagy gond, de ahol 256 vagy még több csomópont van, ott a csomópontok telepítésére és indítására fontos önműködő és jól időzített rendszert létrehozni. A HA-OSCAR-nál a géptelep több szintre osztását fontolgatjuk. Ez a fajta kísérletezés arra is jó, hogy azonosítsuk és felgyorsítsuk a rendszer telepítésének lassú folyamatait; például a LinuxBIOS a szokásos BIOS helyére tehető, és a vas egy kis piszkálgatása után egy tömörített Linux-rendszermag tölthető be a hidegindítás után – így gyorsabb lesz az indulás. A következő OSCAR-kiadás már csoportos adatszórást fog használni, amit több mint ötszáz csomóponton már kipróbáltunk. A telepítés ideje jelentősen csökkent. A HA-OSCAR ezt a módszert az alap telepítési folyamatához át fogja venni és tovább fogja fejleszteni.

Választható csomagok

Az OSCAR telepítéséhez hasonlóan a HA-OSCAR felhasználói is szabadon dönthetnek, hogy melyik alkalmazáscsomagot kívánják telepíteni. Alapértelmezésben a HA-OSCAR önműködően telepíti a géptelep létfontosságú

részeit, majd megkérdezi a felhasználót, hogy milyen alkalmazásokat szeretne futtatni. A telepítési folyamat figyelembe veszi a meglévő beállításokat és a csomópontra már telepített csomagokat. Bizonyos csomagok érzékenyek az adott rendszerkönyvtár – például a *glibc* – típusára. A felhasználóknak tudniuk kell arról, hogy a HA-OSCAR telepítése miatt a rendszerüket esetleg frissíteniük kell, hogy kielégítsék ezeket a függőségeket. Ugyanígy az eltávolítás is megoldott: minden HA-OSCAR-hoz tartozó összetevő törölhető a rendszer épségének (dependencies) elvesztése nélkül. Ez a lehetőség azoknak a felhasználóknak fontos, akik csak ismerkedni szeretnének a HA-OSCAR-ral. Érdemes megemlíteni, hogy a telepítést és eltávolítást végző csomag a v2.0 óta része az OSCAR-alapcsomagnak, és nemsokára kijön egy új, továbbfejlesztett változat.

Frissítés a hálózaton keresztül

A HA-OSCAR fejlesztőcsapata azt tervezi, hogy a rendszer leállítása nélkülmegvizsgálja a programok választható hálózaton keresztüli frissítésének lehetőségét. A hálózati frissítés érdekes módszer az alkalmazások és az operációs rendszer foltozására. A legtöbb Linux-terjesztés manapság önműködő hálózati frissítési lehetőséggel rendelkezik, amely megkönnyíti az unalmas karbantartási munkát. Nagy géptelepek karbantartása esetén a HA-OSCAR felhasználói e képesség segítségével gond nélkül frissíthetik az alkalmazásukat, a szolgáltatások nem szakadnak meg. A hálózati frissítés leegyszerűsíti a géptelep karbantartását, és jobb programkezelést segít elő a csomópontokban. Ráadásul a HA-OSCAR-ban van egy segédprogram, amelylyel a felhasználó úgy változtathatja meg menet közben a géptelep beállításait, mint ahogyan a LinuxConfot lehet használni. Ez még gyerekcipőben jár, de a közeljövőben dolgozni fogunk rajta.

Mentés, helyreállítás, hibaelhárítás

Egy számítástechnikai rendszerben nem lehet megbízni, ha nincs hozzá mentési és helyreállítási terv. A létfontosságú programoknál – például a távközlésben – fontos, hogy a rendszert bármilyen program- vagy géphiba után helyre lehessen állítani. Emiatt minden HA-rendszernek létfontosságú része a mentési és helyreállítási alrendszer.
A végzetes hiba utáni helyreállítás képessége és sebessége nagyon fontos. A HA-OSCAR minden teljes újratelepítése vagy rendszermagjának frissítése előtt és után a lemezképet a mentési kiszolgáló egy megadott helyére és szalagra mentjük. A Ghost for Unix pillanatfelvételt készít a régi és az új rendszermagról, tömöríti azt és elküldi a másodlagos főcsomópontra, valamint a katasztrófa utáni kijelölt helyreállító gépre. A lemezképbe fontos adatok, alkalmazások és beállításfájlok is felvehetők. Általában naponta szalagra mentjük az éjjeli pillanatfelvételek különbségeit, de a teljes lemezképeket csak hetente. A gyorsabb helyreállítás és a megbízhatóbb mentés érdekében lemezképkészítést, állománynaplózást és adatsokszorozást alkalmazunk.

Webes géptelepek támogatása

A HA-OSCAR egyik célja, hogy webkiszolgálóként is telepíthető legyen, és magas rendelkezésre állású webkiszolgáló-telepet lehessen létrehozni nagyszámú ügyfél számára. A cél eléréséhez egy lépés az Apache-nak minden csomópontra való telepítése; az Apache lehet az egyik csomag, amelyet a csomópontokra lehet másolni. Ezután egyetlen IP-címet rendelünk a géptelephez, valószínűleg az LVS közvetlen útválasztás segítségével, mert az bizonyítottan méretezhető megvalósítás.

Aszinkron folyamatvégrehajtás támogatása

A távközlési alkalmazásokat fel kell készíteni a kivételes és a váratlan helyzetekre. Még a hétköznapi helyzetekben is nagy nyomás alá helyezik az előfizetők a szolgáltatókat, magasak az elvárások a rendszer teljesítményével és rendelkezésre állásával kapcsolatban. Az előfizetők nem tűrik el a programok hibáját vagy a hívásuk bizonyos időn túli késleltetését. Ez különösen igaz a távközlés más szolgáltatási területeire, néhány kimondottan valós idejű tulajdonságokat követel. A távközlési alkalmazásokat az előfizetők igényeinek, a programok karbantartási és frissítési költségének, valamint a rendszer rendelkezésre állásának és méretezhetőségének figyelembe vételével kell elkészíteni. A bonyolult osztott programok különleges programozási elveket kívánnak meg. Az évek során bebizonyosodott, hogy a bonyolult rendszerfelületek növelik a hibakereséshez szükséges időt és a programhibák valószínűségét.
Az aszinkron események módszere (AEM) eseményvezérelt fejlesztési módszer, amellyel üzembiztos alkalmazások készíthetők. A rendszereseményekre az alkalmazás a felhasználói térben megvalósított visszahívó függvényre adja a vezérlést. Az AEM-megvalósításban a rendszermag fontos szerepet játszik az események kezelésében és növeli az alkalmazások megbízhatóságát. Emiatt az AEM rugalmas megoldást jelent a programtervezők számára, mert kiterjeszthető keretrendszert nyújt, az új képességeket futási időben hozzá lehet adni a programokhoz, a rendszer vagy a programok újraindítása nélkül. A távközlési szintű követelmények kielégítése érdekében a HA-OSCAR az AEM hatékony támogatását tervezi.

A cikkhez tartozó Kapcsolódó címek megtalálhatóak ➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/OSCAR könyvtárban.

Linux Journal 2003. november, 115. szám

Ibrahim Haddad (Ibrahim.Haddad@Ericsson.com)
Kutató a montreali Ericsson Corporate Kutatórészlegnél, ahol harmadik nemzedékbeli vezeték nélküli IP-hálózatokon dolgozik.

Chokchai Leangsuksun (box@latech.edu)
Docens a Louisianai Műszaki Egyetem Vállalkozás és Információtechnológia Centrumában.Hét évet töltött a Lucent Technologies K+F osztályán, ahol rendszerek megbízhatóságával és magas rendelkezésre állású számítógép- és távközlési rendszerekkel foglalkozott.

Stephen L. Scott (scottsl@ornl.gov)
Vezető kutató az Oak Ridge National Laboratory számítástechnika és matematika osztályán. Alapító tagja az OCG-nek, jelenleg a 2. változat kiadását szervezi. Ezelőtt ő volt az OSCAR projekt munkacsoportjának az elnöke.

Műsorszórás a helyi hálózaton

Bemutatkozik a VideoLAN projekt műsorszóró megoldása.

Bizonyára emlékeznek még a sorozat első részében adott áttekintésre, amelyben az internetes műsorszórás elvi és gyakorlati megoldásait taglaltuk (Linuxvilág 2003. november, 42. oldal: „Műsorszórás az interneten"). Akkor szó esett az egyik legfőbb technikai nehézségről is, amely a multimédiás adatfolyamok továbbításának a sávszélességigényében rejlik. Ez még inkább igaz a videók továbbításának az esetében, amikor a továbbítandó képanyag meghússzorozza a sávigényt, pedig a megfelelő sávszélesség eléréséért még hanganyag esetében is különböző trükkökhöz kell folyamodniuk a rádióállomásoknak.
A manapság bárhonnan elérhető videokiszolgálók igen gyatra minőségben szolgáltatnak, s mindezt csak igen korlátozott néptömegek számára. Ezek alapján teljesen nyilvánvaló, hogy a globális internetes televíziózás Kánaánja még várat magára néhány évet. Ha azonban egy kissé leszűkítjük a kört és csak a helyi hálózatokat (LAN Local Area Network) helyezzük a figyelem középpontjába, máris láthatjuk, hogy ezek bizony az összes szükséges követelményt teljesítik, amelyek közül talán a csoportos módú (multicast) üzenetküldési lehetőség a legfontosabb. Mi lenne tehát, ha kezdetben feladnánk világuralmi törekvéseinket, s csupán a dobbantónak számító helyi hálózaton építenénk ki videoműsorszóró megoldást? A válasz igen egyszerű: minden bizonnyal sikerrel járnánk, s ehhez nem kell semmi mást tennünk, mint használni a VideoLAN projekt által kínált lehetőségeket.

Miről is van szó pontosan?

A manapság elterjedt helyi hálózati topológiák nagyságrendileg 100 Mbites sávszélességet képesek nyújtani. Ehhez vegyük hozzá, hogy a csoportos üzenetküldés lehetővé teszi számunkra, hogy a kiszolgáló a kívánt adatot egyetlen csomag elküldésével minden ügyfélhez egyidejűleg eljuttassa, azaz a hálózaton minden pillanatban pontosan egy adatfolyam halad, pontosan egyszer foglalva le a szükséges sávszélességet. Ha egy kicsit utánaszámolunk, egy 100 Mbites hálózat esetében még a jelenlegi legnagyobb, azaz 10 Mbit/sec sávigényű szabványos MPEG-2-videofolyamot alapul véve is csaknem tízszer beférünk a helyi hálózat adta keretek közé. Ez több, mint elég. Így a kiváló minőségű tévéadás mellett még vígan használhatjuk a hálózatot arra, amire eddig is: dolgozhatunk, másolhatunk stb.

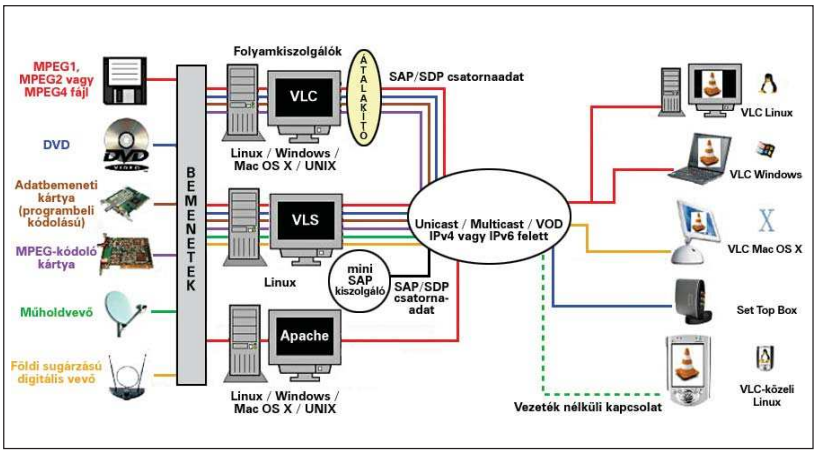
Így működik a VideoLAN

A VideoLAN egy kiszolgálóból és egy ügyfélből áll, ezek fejlesztése párhuzamosan folyik, s jelenleg az ügyféloldali program (VLC) a kifinomultabb állapotú. Ennek az ügyfélnek az az érdekessége, hogy egyben kiszolgálóként és médialejátszóként is működik. A kiszolgáló (VLS) inkább egy erőteljes, de szerény képességű program, amely a távoli irányítás megsegítésére `telnet` felülettel bír.

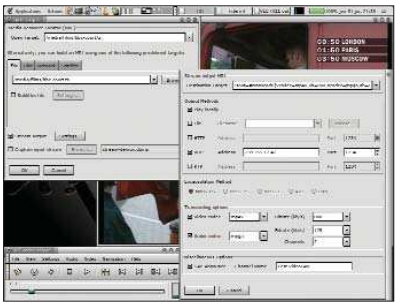
Az adattovábbítási módokat tekintve három protokoll használatára nyílik lehetőségünk: az első a webről már jól ismert HTTP protokoll, amely a legtöbb mai műsorszóró megoldás alapja. A másik a kevésbé ismert RTP (RealTime Protocol), ami kifejezetten valós idejű adatfolyamok továbbítását teszi lehetővé a fejrészbe illesztett médiafüggő időbélyeg segítségével. (Ezzel a két protokollal természetesen nem csak helyi hálózatokra alkalmazható megoldást kapunk.) A harmadik lehetőség maga a program aduásza, az UDP protokoll, amelyet leggyakrabban csoportos módú üzenetküldés során használunk a helyi hálózaton. Ezeknek a protokolloknak a segítségével tehetjük lehetővé az adatfolyam elérését az ügyfelek számára, vagy éppen adatokat lőhetünk ki a hálózatra, amit az ügyfél programja venni képes. (Ezt jobban átgondolva valóban a hagyományos értelemben vett „műsorszóráshoz" jutunk.)
A bemeneti eszközök és formátumok száma rengeteg: a DVB-kártyák adataitól kezdve az FTP-n keresztül letöltött adatfolyamig bármit sugározhatunk, s mindezt a legkülönfélébb formátumokban: MPEG1, MPEG2, MPEG4 (DivX). Itt azonban hozzá kell fűznöm, hogy számos formátum

Az UDP protokoll

Az internet-protokollcsomag része; olyan szállítási protokoll, amely beágyazott IP-datagramok küldését teszi lehetővé összeköttetés nélkül, ugyanakkor nem garantálja a csomagok megérkezését. Akkor van jelentősége, amikor sem sorba rendezésre, sem forgalomszabályozásra nincsen szükség. Elsősorban olyan kiszolgáló alapú alkalmazásoknál terjedt el, amelyekben a gyors adattovábbítás sokkal fontosabb, mint a pontos adattovábbítás, s amelyek a hozzájuk érkező kérésekre ilyen UDP-keretekkel válaszolnak. Jellemzően ilyen alkalmazások a beszéd- vagy videoátvitelt lebonyolító programok.



1. kép A VideoLAN műsorszóró megoldása



2. kép A VLC grafikus felhasználói felülete

vételére csakis a VLC mint ügyféloldali vevőprogram képes, ezért az általános használat során ezt minden olyan gépre telepítenünk kell, amin nézni szeretnénk az adást. A kezelt formátumokról, protokollokról a
➲ http://http://www.videolan.org/vlc/features, illetve a
➲ http://http://www.videolan.org/streaming/features.html oldalakon tájékozódhatunk.
A módszer hátránya mind a továbbítás módját, mind a sugárzott adatfolyam típusát tekintve az, hogy egyetlen elterjedt szabványra sem illeszkedik. Ezen a fejlesztők olyan módon próbáltak meg segíteni, hogy a VLC-ügyfél szinte mindenegyes jelenleg használatos operációs rendszeren (Windows, Mac OS, BeOS, Linux, FreeBSD, Solaris, QNX) elérhető, valamint a hálózatra csatlakoztatva számos set-top box is képes fogni a sugárzott adást. Így a rengeteg vételi lehetőség következtében már megvan az esély arra, hogy igen széles körben elterjedjen. Mindezekhez természetesen olyan kiegészítő szolgáltatásokat adhatunk, mint a SAP/SDP protokollon alapuló csatornainformációs szolgáltatás, amellyel a sugárzott programokról kaphatunk adatokat.
Jelen írásunkban inkább a VLC műsorszóró képességeivel ismerkedünk meg, hiszen sokkal színesebb, s talán a módszer elsajátítására is alkalmasabb, mint a kiszolgáló, amelynek a kezelése meglehetősen nehézkes. Ezen belül is a helyi hálózaton történő tévézést részesítjük majd előnyben, hiszen itt lehet igazán sikerélményünk, s mint tudjuk, ez a továbbiakban nagyon fontos lehet lélektanilag.

Mi szükséges hozzá?

Ahhoz, hogy a cikkben leírtakat ki tudjuk próbálni és felépítsünk egy kezdetleges „televízióállomást” a hozzá kapcsolódó „televíziókészülékekkel”, egy kisebb helyi hálózatra lesz szükségünk, ami két, de inkább három gépből áll, egy csomóponttal (hub) vagy kapcsolóval (switch) összekötve. A legjobb, ha a három gépen legalább két különböző operációs rendszer van. A kiszolgálóra telepítsünk Linuxot, a másik két vevőre pedig Windowst, illetve Linuxot – mindezt csak azért, hogy lássuk a módszer felületfüggetlenségét. Ezek után indítsunk egy adatfolyamot, s tanulmányozzuk, hálózatunk miként viselkedik az új körülmények között.

Csapjunk bele!

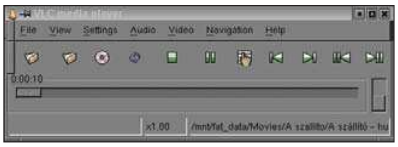
Mielőtt belecsapnánk, nézzük meg, hogyan tudjuk telepíteni a programot. Látogassunk el a
➲ http://http://www.videolan.org/vlc/ oldalra, s keressük meg a megfelelő operációs rendszerekhez tartozó változatot, majd töltsük le a programot vagy járjunk el a leírásnak megfelelően. Érdekes, hogy az oldalon, a Linuxon belül külön-külön találunk telepíthető binárisokat az egyes terjesztésekhez, sőt Debianhoz egyenesen apt-forrást adnak a fejlesztők. A magas színvonalú telepítési támogatás következtében nekünk semmi dolgunk. A rendszer csomagkezelőjével telepítenünk kell az adott csomago(ka)t, s a rendszer futásra kész.

Most már valóban csapjunk bele!

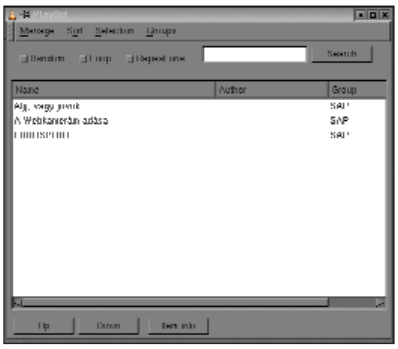
Próbálkozzunk mindjárt a megoldás középpontjában elhelyezkedő UDP protokollra épülő adatfolyam-továbbítással. Szerezzünk be egy videofájlt, s az alábbi parancs kiadásával játsszuk át a helyi hálózat másik gépére:

```
vlc -vvv video.fájl -sout
udp:192.168.0.9
```

Ebben az IP-cím a szomszédos gépre mutat (a szomszéd gépen azonban még ne indítsuk el az ügyfélprogramot). Ha mindent jól csináltunk, akkor sem tapasztalunk semmit. A konzolon megjelenik néhány sor üzenet, amelyből arra következtethetünk, hogy az adás él. Annak érdekében, hogy mindenről megbizonyosodjunk, menjünk oda a hálózati csomóponthoz (hub vagy switch) vagy a gép hátuljához, és figyeljük a forgalomjelző LED-eket. Azonnal látjuk, hogy bár látszólag nem történik semmi, a LED-ek hálózati forgalmat jeleznek, tehát az adatfolyam a célgéphez eljut, de a gép az ügyfélprogram futásának hiányában a kapott csomagokat egyelőre nem dolgozza fel. A jelenségen keresztül jól megfigyelhető az UDP protokollnak az a sajátossága, hogy a csomagok egylövetűek: a kiszolgáló az UDP-keretet elküldi a hálózaton, nem törődve azzal, hogy mi lesz a sorsa (ez egyébként teljesen a hagyományos légköri jelsugárzáshoz hasonlatos), és ez jelen esetben még előnyünkre is szolgál, ugyanis nem kell az esetleges vételi hibákkal vesződnünk; a kiszolgáló csak folyamatosan küldi az adatokat, sorban egymás után. Ezzel nincs is semmi baj, hiszen a helyi hálózaton a véletlen csomagvesztés valószínűsége igen kicsi, és a TCP protokoll által biztosított sorba rendezésre sincsen szükség, hiszen itt a csomagok csak egyféleképpen juthatnak el a célállomáshoz, és az adatok eleve sorrendben továbbítódnak.



3. kép A VLC médialejátszó



4. kép
A helyi hálózaton érzékelt csatornák listája

Az elemzés után itt az ideje, hogy az ügyfélgépen elindítsuk a fogadóprogramot. Ehhez – akár Windows, akár Linux az operációs rendszerünk – a következő parancsot adjuk ki:

```
vlc -vvv udp:
```

Ennek hatására rövid várakozás után bekapcsolódhatunk az adatfolyamba. Mivel a VLC a videoformátumok szinte mindegyikét ismeri, nem kell azon aggódnunk, hogy milyen formában küldjük a kiszolgálóról az adatokat, így bármilyen kodekkel kódolt képanyag megteszi (mindegyik formátumnak ugyanúgy történik a fizikai továbbítása).

Élvezzük többen az adást!

Természetes igényként merülhet fel, hogy ha már sugárzunk, ezt úgy tegyük, hogy az adatfolyam mindenki számára elérhető legyen, s a hálózat se terhelődjön túl. Ehhez a csoportos módú üzenetküldési lehetőséget fogjuk segítségül hívni; ennek hatására az adatfolyamot a kiszolgáló nemcsak az adott gépnek továbbítja, hanem mindenkinek a hálózaton, s mindenki ugyanazt a csomagot kapja meg az adott pillanatban. Ezáltal a hálózatban az adatfolyam pontosan egyszer halad végig, ennélfogva bármennyien anélkül bekapcsolódhatnak a műsorfolyamba, hogy akár egyetlen felesleges bit is keletkezne a vezetékben. Így természetesen mindenki az összes részét megkapja az adatfolyamnak, csak legfeljebb nem „dolgozzák fel” azt, s ettől a hálózat nem lassul le jelentősen, sőt az adás érkezésének a ténye valószínűleg észrevétlen marad. Tegyünk egy próbát, és adjuk ki a következő parancsot:

```
vlc -vvv video.fájl
➥--sout udp:239.255.12.42
```

Mivel a továbbítás mindig mindenkihez történik, az IP-címnek jelenleg annyi a szerepe, hogy adatfolyamunkat a hálózaton azonosítsa az az ügyfelek számára. Most menjünk oda a hálózati csomóponthoz, és ellenőrizzük munkánk helyességét. Azt kell tapasztalnunk, hogy a hálózati kapcsolón minden működik, az összes LED egyszerre villog, tehát mindenki megkapja a csomagot, működik a csoportos üzenetküldés. Ezután az ügyfélprogramot bármelyik gépen elindítva vehető az adás. Itt azonban csatlakozni kell az adott IP által azonosított csoportos üzenetküldési címhez, mert elképzelhető, hogy a hálózaton több aktív csoport is jelen van, s ezek közül nekünk ki kell választanunk, hogy melyik hozzánk érkező adatfolyamot szeretnénk lekezelni. Ehhez a fogadóoldalon adjuk ki a

```
vlc -vvv udp:@239.255.12.42
```

parancsot, és lássunk csodát! Célszerű minden gépen elindítani, így láthatjuk, hogy a nagyobb számú nézősereg semmilyen forgalomnövekedést, terhelésnövekedést nem okoz a kiszolgáló oldalán.

Lépjünk tovább!

Most már működik a csoportos üzenetküldés, bárki hozzáférhet az adatfolyamhoz. Ez így egy egészen jól használható rendszer, ám ami a szolgáltatásokat illeti, még igencsak kezdetleges: eddig csak helyi fájlt tudtunk kiküldeni, s a csatlakozás is nehézkes, nem túl felhasználóbarát. Annak érdekében, hogy ezeket feljavítsuk, először is nézzük meg, hogy milyen adatfolyamot sugározhatunk a hálózaton – a merevlemezen és a CD-n levő fájlokon kívül.

DVD, SVCD, VCD, CDDA

DVD-lemezek lejátszása mellett annak tartalmát a helyi hálózaton elérhetővé tehetjük. Ehhez a lemezt még be sem kell fűznünk, csak tegyük be a meghajtóba, majd adjuk ki a következő utasítást:

```
vlc -vvv dvd:[paraméterek]
➥--sout udp:@239.255.12.42
```

Hasonló módon járhatunk el, ha video vagy super video CD-t szeretnénk lejátszani. Ekkor a `dvd:` tagot `svcd:` tagra kell cserélnünk, és más kapcsolók megadásával szóra bírhatjuk a kiszolgálót. Bár most alapvetően videosugárzásról van szó, meg kell említenem, hogy a fentiekhez teljesen hasonlóan akár audio-CD-ket is kiküldhetünk a hálózatra, de itt bemeneti eszközként a `cdda:`-t kell megadnunk. Természetesen MP3-akat vagy más zenei formátumokat (pl.: ogg) is lejátszhatunk, rádióállomássá minősítve vissza kiszolgálónkat.

Multicast-címek

Csoportos üzenetküldés esetén a csomagokat ugyanúgy címezni kell, de itt egy különleges IP-címet kell megadni célként, s az így megjelölt üzeneteket a hálózaton mindenki megkapja. Továbbmenve, a nagyobb hálózati forgalomirányítókon (router) is beállítható, hogy mit kezdjenek az ilyen üzenetekkel, hová továbbítsák stb. Így egy egész gerinchálózat építhető, amely támogatja a csoportos üzenetküldést, lehetővé téve, hogy ezek a bizonyos csomagok kilépjenek a helyi hálózatból.
A szabvány szerint erre a különleges címzésre a régi D osztályú IP-címeket használják (224.0.0.0–239.255.255.255). Ezek természetesen a különböző előírásoknak megfelelően további tartományokra oszlanak, amely tartományoknak a kiosztása az IANA (Internet Assigned Numbers Authority) feladata. Ezek közül mi találomra bármelyik címet használhatjuk, az otthoni hálózatban nem fogunk semmilyen különbséget észrevenni, ám ha világméretű környezetben használjuk, kénytelenek leszünk a szabványokhoz igazodni.

Megjegyezném, ahhoz, hogy az itt leírt parancsokat alkalmazni tudjuk, a megfelelő eszközökre írási jogosultságra lesz szükségünk – ezt ne felejtsük el előre beállítani.

DVB csatorna

A számítógépbe szerelhető DVB-dekódolókártyák nagy többsége szabványos, ezért Linux alatt is szóra bírjuk, és a segítségükkel digitális tévéadás fogadására nyílik lehetőségünk. Ez gyakorlatilag egy különleges tévétuner kártya, amely kifejezetten digitális adások (DVB-C/S/T) vételére lett kifejlesztve. (Manapság már Magyarországon is elterjedtek a DVB-S-vevők, főleg olyan településeken, ahol kábelrendszer hiányában a lakosok műholdvevő segítségével, digitálisan jutnak hozzá a közel száz elérhető csatornához).
Amennyiben ilyen adást szeretnénk a helyi hálózaton elérhetővé tenni, az eddigiekhez hasonlóan bemeneti eszközként a dvb kártyánkat kell megadni, a megfelelő kapcsolókkal ellátva. Ez ügyben a ➲ http:/www.videolan.org/doc/videolan-howto/en/dvb.html weboldalon tájékozódhatunk.

MPEG-enkódoló kártya

Ez egy újabb bemeneti eszköz, amely arra képes, hogy a rá kötött adatfolyamot valós időben alakítja át a megfelelő MPEG formátumra. Ahhoz, hogy élő tévéadást fogjunk, szükségünk lesz egy ilyenre a rákötött kamerával együtt – ez szolgáltatja a bemenő képanyagot. MPEG-kártyánkat megfelelően telepítve a VLC bemeneti eszközének ezután ezt kell forrásként megadni. A forrás pontos beállítása meglehetősen bonyolult, terjedelmi okok miatt nem részletezem a cikkben, ám ha valakit érdekel, akkor a ➲ http://www.videolan.org/doc/videolan-howto/en/hardencoding.html oldalon utánanézhet.

Egyéb lehetőségek

Ezen túl még a v4l-megfelelő eszközökről (tévétuner, webkamera) is képet nyerhetünk, s ezt átkódolva (transcode) szintén továbbküldhetjük a hálózatra. Megjegyzendő, hogy ilyen esetekre egy meglehetősen erős masinát kell rendszeresítenünk, olyat, amelyik képes ellátni a videotömörítés gépigényes feladatát. Ha valakit érdekel, akkor a ➲http://www.videolan.org/doc/videolan-howto/en/softencoding.html oldalon talál bővebb leírást a módszerről.
Most, hogy tisztában vagyunk a bemeneti eszközök sokaságával, itt az ideje, hogy némi minőségbeli többletet vigyünk az adatfolyamba. Módunkban áll például a sugárzott adatfolyamról információt küldeni az ügyfeleknek. Ezt kétféleképpen tehetjük meg: a VLC kezdetleges, de egyszerű csatornainformációs szolgáltatásával, vagy a Mini-SAP kiszolgáló segítségével, amit azonban csak a VLS-sel tudunk együttműködésre bírni. Mindezt a szabványos SAP/SDP (Session Advertising Protocol/Session Description Protocol) protokoll használatával valósíthatjuk meg. A módszer eredményeképpen az adatfolyammal párhuzamosan az adott műsorról rövid leírás is továbbítódik a hálózaton, amelyet az ügyfél felismer, s önműködően hozzáadja a lejátszási listához, így nekünk elég a vevőoldalon elindítanunk a VLC-ügyfelet, s azonnal látjuk, hogy éppen milyen műsorok foghatók a hálózaton. Az adott lejátszási listaelemre kattintva pedig feliratkozik a kívánt csoportos UDP-folyamra, és láthatóvá válik az adás. A gyakorlatban a kiszolgálóoldalon meg kell mondanunk, hogy a SAP protokoll használatával az alábbi paranccsal információt fűzzön az adatfolyam mellé:

```
vlc -vvv video.fájl --sout
➥'#standard{access=udp,
➥url=239.255.12.42,sap=
➥"Próba adatfolyam"}'
```

Az ügyféloldalon ilyenkor a VLC-t a `--extraintf sap` kapcsolóval kell indítanunk. Ha ekkor megnézzük a lejátszási listát, látni fogjuk benne a „Próba adatfolyam” nevű adást, amelyet egy dupla kattintással meg is nézhetünk.

A VLC médialejátszó képességei

Úgy érzem, adós maradnék, ha nem jegyezném meg, hogy a VLC milyen jól használható médialejátszóként. Ismeri a legtöbb fájlformátumot, hálózati forrást (FTP, HTTP) is le tud játszani, így elérhetőek például a népszerű TotalCar autósműsor webről nézhető epizódjai; s a telepítése gyakorlatilag egyetlen paranccsal történik. Ezeken kívül DVD, VideoCD, AudioCD, digitális tévéadás, analóg tévéadás lejátszására képes, ha a gépünkben a megfelelő eszközök rendelkezésre állnak. A lejátszóprogram grafikus felhasználói felülettel rendelkezik, amelyről az összes ügyféloldali szolgáltatás elérhető, nem szorulunk a parancssor használatára. Természetesen nem tud olyan sokat, mint mondjuk az MPlayer, de a Windows alatt elérhető Windows Media Playerrel már nyugodtan versenyre kelhet.

Végszó az eddig leírtak ügyében

A SHOUTcastról szóló cikkben (Linuxvilág 2003. decemberi számának 38–41. oldalak) egy kollégiumi rádióállomás körvonalait vázoltam fel annak érdekében, hogy érzékeltetni tudjam a módszer hasznosságát. A példánál maradva a feladatot egy kollégiumi videostúdióra is kiterjeszthetjük: fejünkbe vehetjük például azt, hogy egy – a lakók által elérhető – digitális videohálózatot valósítunk meg.

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

A VLC-modulok

Mint a legtöbb program, ez is modulokkal dolgozik. Így van ez akkor is, ha csak egyetlen fájlt játsszunk le, de ez a felhasználó számára áttetsző módon történik. Lehetőségünk van azonban előírni az egyes kimeneti modulok használatát, s közben a beállításaikat az alábbi módon megadni:

```
--sout '#modul1{jellemző1=
➥...,jellemző2=...}:
➥#modul2{jellemző1=...,
➥jellemző2=...}:...'
```

A formátum használatával egy új beállítási módhoz jutunk, s minden esetben, amikor bonyolultabb szolgáltatásokat szeretnénk megvalósítani, ilyen formában kell utasítanunk a VLC-t, mert csak így van módunk a különleges beállítási lehetőségek elérésére.

Debian otthonra (5. rész)

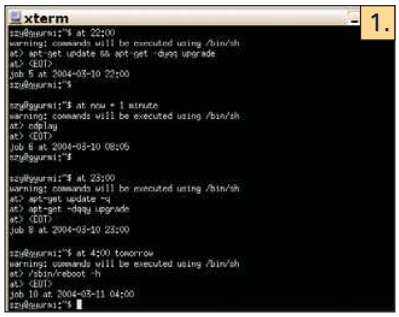
Mélyítsük el rendszergazdai ismereteinket és készüljünk fel lelkiekben a nyomtatásra!

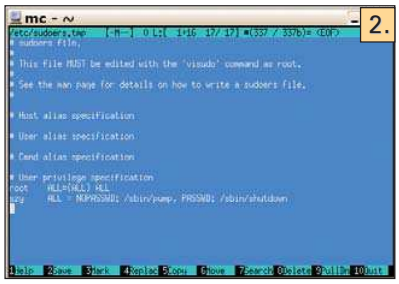
Sorozatunk ötödik részében egy fontos témára, a nyomtatásra készülünk fel. Sajnos a téma elég nagy ahhoz, hogy ne férjen el egy részben, így először lássunk egy kis elméletet!

Először azonban frissítsünk, mint mindig. A szokásos `apt-get update && apt-get upgrade` paranccsal, vagy az Aptitude-ban az `u` és a `g` billentyűkkel. Az előbbi megoldásban javasolt && karaktereket már valószínűleg többször láttad, ez a logikai ÉS művelet. Ezt a trükköt szoktuk alkalmazni, ha egymás után szeretnénk parancsokat kiadni, de olyan módon, hogy ha valamelyik parancs hibával áll le, megszakadjon a folyamat. A két `apt-get` parancsot ezzel a művelettel összekapcsolva például nem szükséges megvárnunk, míg az első befejeződik, majd kézzel elindítanunk a másodikat. (Egy csomaglista-frissítés a kapcsolatunktól függően akár hosszúra is nyúlhat.)

Tény, hogy a megoldás nem tökéletes, hiszen a második program a felhasználóbarátság jegyében már induláskor megáll egy pillanatra, hogy feltegyen egy „Biztos vagy benne?” típusú kérdést. A történet másik fele, hogy a letöltések után a legtöbb rendszergazda örömmel nézegeti, hogy melyik csomag milyen változtatásokat hajt végre. Az volna a legjobb, ha a programot elindítva csak letöltené a szükséges csomagokat, mi pedig majd az esti film után lecsücsülünk a gép elé és óvó tekintetünk kereszttüzében telepítjük az új csomagokat.

A Unix-idők alatt ezek a kérések természetesek voltak, hiszen egy-egy öreg masina a feladataival sokszor órákig, napokig elszöszölt. Erre a célra az `apt-get` is indítható különböző kapcsolókkal. A sorozat első részében már kettőről írtam: ha csak letölteni szeretnénk, egyszerűen a `-d` kapcsolót kell megadnunk; s ha azt szeretnénk, hogy ne tegyen fel „Tényleg ezt akarod?”-típusú kérdéseket, akkor pedig a `-y`-t. Részletes leírás természetesen az `apt-get` súgóoldalán található (`man apt-get`). Ha egy kicsit pontosítjuk a parancsot, akkor nyugodtan felügyelet nélkül hagyhatjuk: `apt-get update && apt-get -dy upgrade`, később pedig az `apt-get upgrade` paranccsal ténylegesen fel is telepítjük a csomagokat. Rendben, közben elmegyek kávézni, de nem lehetne megoldani, hogy megadott időben vagy a megadott rendszerességgel végezze el ugyanezt a gép? És máris elérkeztünk a `cron` és az `at` parancsok világába. Röviden összefoglalva: a `cron` feladata a rendszeresen ismétlődő feladatok elvégzése, az `at`-tal pedig egy adott időpontba tolhatjuk egy parancs (parancsfájl) végrehajtását. Most nem térek ki e két eszköz részletesebb ismertetésére, rengeteg helyen található hozzájuk remek leírás (például Marcel Gagné „Linux rendszerfelügyelet” c. könyvében, a 307. oldaltól kezdődően). Az `at` használata esetén az eredményt levélben kapjuk meg, amiből két dolog következik. Az első, hogy szükségünk lesz egy levéltovábbító rendszerre, a másik pedig, hogy érdemes szűkszavú programokat használni. Az első gondot a rendszer leveszi a vállunkról (az `at` telepítése közben igény szerint felrak egy `courier`-t), a második pedig egyszerűen megoldható, ha a programot „csendes” üzemmódban indítjuk (ezt az `apt-get` esetén a `-q`-val érhetjük el). Ha nincs a gépen `at`, telepítsd, majd add ki az *1. képen* látható parancsokat. Gyakorló rendszergazdák egyből





látják, hogy mi is a baj a képpel: a parancsokat nem rendszergazdaként indítottuk, így a rendszer nem lesz hajlandó frissíteni a csomaglistát, de még a gépet sem fogja hajnali négy órakor leállítani. Egyszerű felhasználóként legfeljebb a `cdplay`-t használhatjuk (ha a felhasználó szerepel a ***cdrom*** csoportban). Mindegy, egy kellemes Jimmy-lemezzel ébresztőórát és szomszédidegesítőt így is készíthetünk. Sokszor kerül az ember az előzőhöz hasonló helyzetbe, amikor is remek volna, ha egyes programokat egy egyszerű felhasználó is rendszergazdai jogkörrel futtathatna, de mégsem volna bölcs kiadni a legfontosabb jelszót. Hordozható gépemen például a hálózati beállításokkal kapcsolatban volna szükség rá, ugyanis a gépet két helyen használom, az egyik helyen rögzített IP-címet kell beállítanom, a másik helyen viszont a DHCP-kiszol-

gálótól kell lekérni az adatokat. Debian alatt található egy `pump` nevű program, ami vígan elvégzi az utóbbi feladatot, tehát csak ezt az egyetlen parancsot szeretném néha rendszergazdaként futtatni. Itt lép színre a `sudo`. Feladata, hogy a `su`-hoz hasonlóan rendszergazdai jogokat adjon, de csak az adott programok futtatása erejéig. Anélkül, hogy elkezdenénk a biztonsággal foglalkozni: a */etc/sudoers* állományban adhatjuk meg, hogy ki milyen parancsokat hajthat végre rendszergazdai joggal. Ezt a fájlt a `visudo` paranccsal szerkesszük! Ezek után az engedélyezett parancsokat a `sudo` ***parancs*** formában hajthatjuk végre.

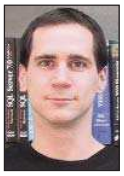
No de, ha már az elméleti anyagoknál tartunk, járjuk körbe egy kicsit a nyomtatás körüli mizériát. Előbb-utóbb rá kívánjuk venni majd gépünket, hogy nyomtasson. Ez fontos feladat, főleg ha irodai munkára akarjuk használni a Debian Linuxot.

Nyomtatás – egy kis elmélet

Hmm. Kezdjük a rossz hírrel. A nyomtatásvezérlők világában is elég nagy a káosz. A helyzetet színesíti, hogy a nyomtatógyártók sokszor még egymás között is nagy vitákat folytatnak, de leginkább nem is szólnak egymáshoz, hanem – a piac törvényeit követve – egymás ellen dolgoznak. Mindegyik igyekszik feltalálni a spanyolviaszt, újabb és újabb nyomtatóvezérlő leíró-nyelvek készítésével. Így aztán szinte mindegyik nyomtatónak más vezérlőprogramra van szüksége. No, ezzel el is érkeztünk a gondok forrásához. Régen egyszerűbb volt az élet, a mátrixnyomatók nemigen dolgoztak színekkel, vonalakkal és képekkel. Olyan volt ez, mint a LOGO nyelvben a teknőc – alapparancsokat megért: ***fel, le, fordul, csúnyán néz***, és itt vége is a tudásnak. Hogy a szöveget éppen aláhúzza, esetleg kisebb, vagy nagyobb betűkkel dolgozzon – ezt egyetlen egyszerű vezérlőnyelvvel meg lehetett vele értetni. Ez az ESC/P vezérlőnyelv volt, a mai napig is sokan használják. A lényege, hogy egy ESC karakter után küldi ki a parancsot, a szöveg közben. Később, amikor már grafikákkal is kellett dolgozni, fontosabb lett a pontosság, a képhűség, rengeteg nyelv jött létre. Kettőt emelek ki: az egyik az Adobe gyermekeként született PostScript, a másik a HP által fejlesztett PCL. A PostScript gépfüggetlennek készült, elsősorban a hordozhatóság jegyében, így általában elmondható, hogyha egy nyomtató tud „postscriptül”, akkor nyert ügyünk van. A mai nyomtatók, hogy tovább bonyolítsák a helyzetet, több nyelvet is beszélnek, több-kevesebb sikerrel. Előfordulhat például, hogy egy nyomtató ugyan beszél „postscriptül és PCL-ül” is, de a saját vezérlőjével lényegesen jobb nyomtatási minőséget tud nyújtani. Természetesen ezt az őrültekházát nem lehet könnyedén egységesen kezelni. A Unix alapú rendszerekben nagy hagyománya van a PostScriptnek, így viszonylag könnyen tudunk ***.ps***-fájlokkal (nyers PostScript-állomány) dolgozni. Ezért az az egyik lehetséges megoldás, hogy a nyomtatni akaró programot felkészítik ***.ps***-ek készítésére, majd a kimenetet egyszerűen elküldik a nyomtatónak. A PostScript nagy előnye ugyanis, hogy nem kell hozzá folyamatos kapcsolat: a fájl valahogy átjut a nyomtatóhoz, a nyomtató értelmezi azt és elkészíti az oldalt. Hátránya viszont, hogy így a nyomtatóra hárul az oldal értelmezése, ami gyakran komoly feladat. Régebben azok a nyomtatók, amelyek rendes PS-értelmezővel voltak felszerelve, gyakran másfélszerkétszer annyiba kerültek, mint PS-támogatás nélkül rendelkező testvéreik. Pont ezt a nagy árkülönbséget kívánták meglovagolni a „winprinter” néven elhíresült nyomtatók. Ezek nem rendelkeznek saját értelmezővel, sőt lényegében semmilyen komolyabb vezérlőegységgel, így harmad-negyedáron lehet őket piacra dobni. Cserébe viszont folyamatosan a gépnek kell vezérelnie őket. Ez a régi számítógépeknél olyan komoly megterhelést jelentett, hogy a munka szinte megállt a nyomtatás idejére. Képzelj el egy közértet, ahol a pénztárgépek megállnak nyomtatáskor! Egy szó mint száz, ezeket a processzor nélküli nyomtatókat az egész szakmának volt ideje megutálni.

A másik elmélet, amivel a nyomtatókat megkísérlik „egységesen” kezelni, szintén komoly megterhelést jelent a gépnek, de jó eséllyel megoldja ezt a gondot. Ez pedig egyszerű: a nyomtatónak minden anyagot képként küldjünk ki! Ha tudjuk, hogy a nyomtató legfeljebb 300×300 dpi-vel képes nyomtatni, készítünk egy papírméretű 300×300-as felbontású képet, és azt küldjük ki. Igen, ez is működik, de szintén vannak hátulütői: az értelmezés sok időt vehet el a géptől, az elkészített kép rendkívül nagy, valamint nem utolsó sorban általában nem ismerjük a nyomtató képességeit. Nem is beszélve arról, hogy a nyomtatók általában lényegesen szebben írnak, ha csak annyit mondunk nekik, hogy „ide írj egy ekkora és ilyen családba tartozó g betűt”.

Mint látjuk, nem egyszerű a feladat, sőt hogy még egy lapáttal rátegyünk, Linux alatt is több nyomtatóvezérlő program létezik. Közülük kettő használatos mostanában széles körben, az `lprng` és a `cups`. A cikk következő részében ezekkel foglalkozunk, majd végre tényleg kikövetelünk valamicske eredményt.



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

Tíz tipp az OpenOffice.org-ra való áttéréshez

Egy irodai programcsomaghoz hasonlóan bonyolult alkalmazás lecserélése komoly vállalkozás. De ha néhány, az igényeinket biztosító, egyszerű szabályt betartunk, az új programmal is rögtön el tudjuk látni a feladatainkat.

Az elmúlt másfél évben azzal nyaggattam az embereket, hogy próbálják ki az OpenOffice.org-ot (OOo). Lassan rájöttem, hogy nem elég csak beszélni a szabad programokról. Nem arról van szó, hogy az embereknek nem tetszik az ingyenes letöltés gondolata, vagy nem szeretnének egy projekthez csatlakozni és beleszólni annak a fejlesztésébe. Vajon akkor mi a baj?
Az, hogy a szabad programok szeretete a legtöbbször nem elegendő ahhoz, hogy zökkenőmentesen áttérjünk az OpenOffice.org-ra. A következőkben az átállást segítő javaslataimat olvashatják.
Még a legnyitottabb gondolkodású embernek is megfordul a fejében, hogy felhagyjon a próbálkozással. Ennek ellenére végezzük el az előkészületeket, és próbáljunk ki néhány szolgáltatást. A legfontosabb, hogy szánjunk rá néhány órát, és használjuk, mielőtt eldöntenénk, hogy OpenOffice.org megfelel-e az az igényeinknek. Nem lehet segíteni azoknak, akikből hiányzik a kíváncsiság vagy a felfedezés iránti vágy, de ha a következő tanácsokra odafigyelünk, az egész sokkal könnyebben fog menni.

1. Ne a hiányzó szolgáltatásokat keressük!

Az emberek gyakran azt mondják, hogy „szeretném az OpenOffice-t használni, de…”, és megemlítenek egy számukra életbevágóan fontos szolgáltatást. Néha csak rápillantanak az OpenOffice.org-ra, és azt mondják, hogy az a bizonyos lehetőség hiányzik belőle. A legtöbbször meg tudom nekik mutatni, hol találják meg; ekkor zavart csönd következik, és témát váltanak. Előfordul, hogy tényleg azt gondolják, hogy az adott szolgáltatás hiányzik, de ez egyre gyakrabban csak kifogás. Bárhogyan is van, gyanítom, hogy az a feltételezés áll emögött, hogy a szabad program mindig gyengébb minőségű, mint kereskedelmi megfelelője. Ezt az előítéletet erősíti, ha az adott szolgáltatást valamilyen félvállról vett keresés alkalmával nem találjuk meg azonnal.
Még ha valós is a gyanúm, a Linux-, Apache-, Gimp- és Mozilla-felhasználók mind tudják, hogy az említett feltételezés nem igaz, az OpenOffice.org-ra pedig semmiképpen nem az. Annak ellenére, hogy írásom születésének a pillanatában az OOo az 1.1-es változatnál tart, a Sun tulajdonát képező StarOffice hosszabb történetet tudhat maga mögött, mint azt a legtöbben gondolják. A StarOffice első változata valójában egy DOS-os szövegszerkesztő volt, ami 1985-ben jelent meg. Két évtizednyi fejlesztéssel a háta mögött az OpenOffice.org kódja kiforrottnak és általában befejezettnek tekinthető. Kétségtelen, hogy más szövegszerkesztőknek vannak olyan szolgáltatásai, amelyek az OpenOffice.org-ból hiányoznak. Nincs meg benne az MS Word helyesírás-ellenőrzője (grammar checker), a FrameMaker mesteroldal (master pages) tulajdonsága. Ezzel szemben az OOo-nak is léteznek olyan jellemzői, amelyek ezekből a vetélytársakból hiányoznak.
A lényeg az, hogy ne borúlátóan tekintsünk az OOo szolgáltatásaira. Általában bízhatunk abban, hogy az adott szolgáltatás fellellhető valahol a menüben. Lehet, hogy egy kicsit más formában, mint amit megszoktunk – például az OOo vázlatkészítő eszközének működése erősen eltér az MS Word eszközétől –, de a különbségek ellenére valamilyen más formában megtalálhatók.

2. Ne várjuk el, hogy a szolgáltatások ugyanott legyenek!

Nem kétséges, hogy az OOo felülete hasonlít az MS Office felületéhez. Az általános menüszerkezet is gyakorta megegyezik, beleértve azt a zavaró dolgot is, hogy egyazon menüben található a ***Testreszabás*** (Configure) és a ***Beállítások*** (Options) menüpont.
A felhasználói felületek hasonlósága megkönnyítheti az OOo-ra való áttérést, ugyanakkor félrevezető is lehet. Helyenként az OOo rendet rakott és ésszerűsítette az MS Office menüszerkezetét. Például a táblázatok nem kaptak külön menüt az OOo-ban; ehelyett a ***Beszúrás*** (Insert) menübe kerültek. Más esetekben pedig ugyanannak a szolgáltatásnak eltérő a neve: például az MS Word ***Autosum***-ja az OOo-ban ***Automatikus összefoglaló*** (AutoAbstract) néven található meg.
Más szóval az OOo a megszokott és az új dolgok keveréke. Szerencsére a legfontosabb szolgáltatásokat rendszerint a megszokott helyükön találjuk, így a tapasztalatlanabb felhasználók sem tévednek el egykönnyen. De a járatosabbaknak is nagyobb fokú rugalmasságra lehet szükségük. Ha egy eszközt nem találunk a feltételezett helyén, akkor próbáljuk meg

kitalálni, hogy milyen más menüben kaphatott helyet vagy hogyan nevezhetik másképpen. Ha a képzelőerőnk nem elegendő, nézzük meg a ***Súgó*** (Help) állományok ***MS Office-jellemzők összehasonlítása*** (MS Office Feature comparison) menüpontot vagy a saját részletesebb összehasonlításomat a ➲ http://www.raycomm.com/techwhirl/magazine/technical/openofficewriter.html címen. Többnyire megleljük, amire szükségünk van.

3. Ne gondoljuk, hogy hosszas képzésre lesz szükség!

Bár néhány eszköz lecserélése nehézséget okozhat, valószínűleg nincs szükség hosszú áttérési időre, mielőtt a cégünk vagy mi magunk az OOo-ot hatékonyan tudnánk használni. Egy hét valószínűleg elég az áttérésre.
A rendszer képességeit legkevésbé kihasználó felhasználók számára az áttérés nagyon egyszerű, mert ők a szövegszerkesztőt gyakorlatilag írógépként veszik igénybe. Ha egy szót dőlt betűvel szeretnének szedni, nem a ***Hangsúlyozás*** (Emphasis) karakterstílust használják, hanem kijelölik a szót, és a dőlt betű ikonra (italic icon) kattintanak. Ha később úgy döntenek, hogy mégis inkább félkövér karakterekkel hangsúlyozzák a szavakat, végignézik a dokumentumukat, és minden egyes szó formázását külön-külön módosítják. Ezek az alapvető módszerek egyik szövegszerkesztőben sem hatékonyak, nem is beszélve az OOo-ról. Az így dolgozó emberek azonban csak kevés eszközt használnak. Az OOo-ban ezeket a szolgáltatásokat általában a megszokott helyen találjuk. Például a karakter- és a bekezdéstulajdonságokat a ***Formátum*** (Format) menüben és a tálcán, a helyesírás-ellenőrző pedig az ***Eszközök*** (Tools) menüben szerepel Az OOo-ra történő áttérés jó alkalom lehet a tanulásra, de a felhasználók közben napi feladataikat szinte teljesen zavartalanul végezhetik.
Lehet, hogy a tapasztaltabb felhasználóknak egy-két nappal több időre van szükségük, hogy megszokják a rendszert. Mindazonáltal éppúgy, ahogyan az egyik nyelv ismerete segíti egy hozzá hasonló másik nyelv elsajátítását, egy szövegszerkesztő ismerete is segít egy másik megtanulásában.
A tapasztalt felhasználók határozott elvárásokkal bírnak, és általában elégé magabiztosak, hogy saját maguk keressék meg a megoldást, ezért nekik nincs szükségük az OOo-oktatásra – maguktól tanulják meg, amit kell.

4. Ne alapozzunk az export- és import szűrőkre!

Első látásra OpenOffice.org tökéletesen alkalmasnak tűnik arra, hogy más irodai programcsomagok – különösképpen az MS Office – dokumentumaival elboldoguljon. A kötegelt konvertálóval (batch converter, helye: ***Fájl/Tündér/Dokumentumkonvertáló*** – File/AutoPilot/Document Converter) többféle MS-formátumba is lehet menteni; sőt az ***Eszközök/Beállítások*** (Tools/Option) menüpontban azt is beállíthatjuk, hogy az OOo alapértelmezés szerint MS formátumba mentsen és megőrizze a számára használhatatlan VB parancsfájlokat. Mi másra lehet még szükség? Először is türelemre, másrészt sok-sok szabadidőre.
Az az igazság, hogy még soha sem készült egyetlen irodai programcsomaghoz sem teljesen megbízható import- és exportszűrő. Igen valószínű, hogy ilyen soha nem is lesz. De ha mégis, én az OOo, a KOffice, illetve a Gnome office együttműködésére fogadnék. Ezeknek a formátumoknak mindegyike nyílt forráskódú, így legalább a fejlesztéshez kevesebb idő szükséges. De a szűrők még a nyílt forrású formátumok esetén is gondot fognak okozni a közeljövőben.
Nem elhanyagolható az a tény sem, hogy egy csomó szűrő a szabadalmaztatott formátumokkal kapcsolatos.
Ez azt jelenti, hogy a fejlesztőknek az adott formátumot vissza kell fejteniük, ami nehéz, időigényes és esetenként jogilag is kockázatos folyamat. Az MS Office-szűrők elkészítése mint a legfontosabb feladat különösen nehéz, mivel a formátum gyakran változik, és sokszor még a saját korábbi változataival sem működik együtt.
Az OOo natív XML formátuma megkönnyíti a szűrők írását, és az OOo MS Office-szűrője az egyik legjobb, amit valaha láttam. De még ezek a szűrők is messze állnak a tökéletestől, és azoknak a felhasználóknak, akik ezekre támaszkodnak, bele kell nyugodniuk a kézi újraformázás szükségességébe. Ha az OOo mellett más irodai programcsomagok használatához ragaszkodunk, az átalakítást igénylő dokumentumok esetében lehetőség szerint a rövid, egyszerűen formázott dokumentumokra szorítkozzunk. Keressük meg az OOo ***Súgó***jában a ***Microsoft Office dokumentumok konvertálása*** (About Converting Microsoft Office Documents) részt, amelyben megtaláljuk a kerülendő formázási elemek felsorolását. Jobb eredményt kapunk, ha csak stílusokat használunk és biztosítjuk, hogy mindkét programcsomag hozzáférjen ugyanazokhoz a betűtípusokhoz. Még így is előfordulhat, hogy a legegyszerűbb felsorolások is csonkítva jelennek meg. A hibák elkerülése érdekében érdemes feljegyezni azokat a formázó műveleteket, amelyeket nyugodtan használhatunk.
Ha egy másik irodai programcsomag felhasználói nem akarják az adott dokumentumot szerkeszteni, válasszuk a ***Fájl/Exportálás PDF-be*** (File/Export as PDF) menüpontot, és PDF formátumban adjuk át a dokumentumot.
A PDF-állományok közel állnak a nyílt szabványokhoz, ezért erre a szűrőre lehet alapozni.
Az irodai programcsomagok közötti

dokumentumcserére valószínűleg a HTML és az egyszerűsített DocBook formátum a legalkalmasabb. A mai böngészők mindkettőt meg tudják jeleníteni, és ha máshogyan nem is, szöveges állományként a szövegszerkesztők mindenképpen meg tudják nyitni. Még jobb, ha a cégünket vagy környezetünket rávesszük, hogy csak az OpenOffice.org-ot használja.

5. Írjuk össze, hogyan kell a legalapvetőbb feladatokat elvégezni!

Mielőtt áttérnénk az OOo-ra, írjuk össze, hogy milyen alapvető tevékenységeket végzünk az irodai programcsomaggal. Ez a felsorolás lehetőleg ne tartalmazzon húsz tevékenységnél többet. Írjuk fel cédulákra, hogyan lehet ezeket a feladatokat elvégezni, és mindenkinek adjuk oda a cédulákat. Ha már az összes kolléga magabiztosan oldja meg alapvető feladatait, az első kártyát cseréljük ki valami kevésbé egyszerű tennivaló utasításaira. Napokon, de lehet, hogy csupán órákon belül azt fogjuk tapasztalni, hogy már senki sem használja a kártyákat.

6. Használjuk a Súgót!

Az OOo fejlett súgórendszerrel rendelkezik. A korábbi változatokban a súgóállományokból gyakran hiányzott a környezet leírása és sokszor a szolgáltatásokat sem magyarázta el megfelelően. Ennek ellenére az 1.1-es változattól kezdve a ***Súgó*** valóban értékes részét képezi a rendszernek.
Az áttérés kezdetén mindenki megnézi az első négy hivatkozott oldalt az ***Üdvözöljük az OpenOffice.org Writer súgójában!*** A hivatkozások jó áttekintést adnak ahhoz, hogy mélyebben megismerjük a programot. Az első néhány héten érdemes megfeledkezni a bosszúságokról és engedélyezni az induláskor megjelenő tippeket és a segédet.

7. Kezdjünk a Tündérrel!

Az OOo új felhasználóinak egy sor varázsló is rendelkezésére áll, ezek végigvezetnek az olyan alapvető dokumentumok beállításain, mint a levelek vagy a feljegyzések. Ha a végeredmény nem is felel meg pontosan kívánalmainknak, az OOo-val végzett munka elkezdésének gyors lehetőségét kínálják. Nézzük meg, mit rejt a ***Fájl***/***Tündér*** (Files/AutoPilot) menüpont. Az is fontos, hogy a ***Tündér*** lépéseit összehasonlítsuk a végeredménnyel. Ez kiváló módja annak, hogy megismerjük az irodai programcsomagok, ez esetben az OOo lehetőségeit.

8. Tanuljuk meg a stílusok használatát!

Ha a felhasználóknak abba a csoportjába tartozunk, akik egyenként alkalmazzák a szövegformázó utasításokat, az OOo-ra való áttérésünket használjuk fel arra, hogy megtanuljuk a stílusok használatát. Stílusok segítségével bármely szövegszerkesztőben időt takaríthatunk meg, mert lehetővé teszik, hogy a formázás módosítását egy helyen végezzük el és hogy a módosítások a teljes dokumentumon érvénybe lépjenek. A stílusok az OOo-ban azért is különösen fontosak, mert nemcsak bekezdésekre és karakterekre kínálnak mintákat, hanem oldalakra, szövegkeretekre és listákra is. Ezzel nemcsak a fellépő nehézségek számát csökkentjük, de a munkánk hatékonyságán is javítunk. Az OOo-ban a stílusok használatának kulcsa a ***Stílussegéd*** (Stylist), amely a ***Formátum***/***Stílussegéd*** (Format/Stylist) menüpontban elérhető úszó paletta (floating palette). A ***Stílussegéd***-del gépelés közben gyorsan válogathatunk a stílusok között, módosíthatjuk a már meglévőket és újakat is létrehozhatunk. A stílusok listáját különféle szűrőkön keresztül mutatja meg, így a keresett stílusokat könnyen megtalálhatjuk.

9. Ismerjük meg a Navigátor használatát!

A ***Navigátor*** (***Szerkesztés***/***Navigátor***; Edit/Navigator) szintén egy úszó paletta. A ***Stílussegéd*** palettájához hasonlóan ennek is kulcsfontosságú szerepe van az OOo hatékony használatában. Nevének megfelelően a ***Navigátor*** egyik feladata, hogy segítse a felhasználót a dokumentum különböző részei közötti mozgásban; segítségével táblázatok, OLE objektumok vagy oldalak – a dokumentum szinte – bármelyik elemére ugorhatunk. A dokumentum elemei a létrehozáskor sorszámot kapnak, de ha beszédes neveket rendelünk hozzájuk, a ***Navigátor*** megjeleníti őket, így még könnyebbé válik az elemek közötti váltás.
Mindezek ellenére ne hagyjuk, hogy az elnevezés félrevezessen bennünket. A ***Navigátor*** lényegesen több, mint a dokumentumunk térképe. Ha a ***Címsor***-ra (Heading) ugrunk, a ***Navigátor*** vázlatkészítő eszközzé (outlining tool) válik, amelynek segítségével egész fejezeteket mozgathatunk, és az egér mozgatásával a címsorok szintjét változtathatjuk. Nyissunk meg egy ***Fődokumentum***-ot (Master Document), ez lesz a tartalomjegyzék. A ***Navigátor***-ral akár ***Emlékeztető***-ket (Reminder) is hozzáadhatunk a szöveghez.

10. Keressük meg a rejtett szolgáltatásokat!

Minden programban találhatunk váratlan szolgáltatásokat és billentyűkombinációkat. Ezeket nem igazán meglepetésnek szánják, inkább félig-meddig rejtve maradnak, mert nemigen említik meg vagy emelik ki használatukat a ***Súgó***-ban. Én például gyorsan megtaláltam a ***Szerkesztés***/***Visszavonás*** (Edit/Undo) parancsot. De mivel a legtöbbször a menüt vagy a billentyűzetet használom, hetek múlva fedeztem csak fel, hogyha a tálcán kiválasztom a ***Visszavonás*** (Undo) gombot, kiválaszthatom a visszavonásnak azt a szintjét, ahová vissza szeretnék térni. Ehhez hasonlóan, ha azt akarjuk, hogy egy stílus kiválasztásakor valamilyen szöveget önműködően szúrjon be, használhatjuk a ***Beállítások*** fül (Options) ***Mezők beszúrása*** (Before field) ikonját, hogy a listastílusokhoz bekezdésstílust kapcsoljunk. Ezek után, ha az adott stílust használjuk, a ***Mezők beszúrása*** pontban megadott szöveg anélkül jelenik meg, hogy be kellene gépelnünk. Az ilyen meglepetések három dolog miatt fontosak: önbizalmat adnak a program ismeretével kapcsolatban, új dolgok megtanulására ösztönöznek és gyorsítják a mindennapos feladatok elvégzését. Megéri megkeresni őket.

Linux Journal 2004, 119. szám

Bruce Byfield
A Stormix Technologies termékmenedzsereként és a Progeny Linux System értékesítési és kommunikációs igazgatójaként dolgozott. A Maximum Linux segédszerkesztőjeként is tevékenykedett, és eredetileg ő írta a Desktop Debian-kézikönyvet.

Hatásos: OpenOffice.org Impress

A megfelelő bemutatókészítő program kiválasztása csupán az első lépés egy sikeres beszámoló elkészítésében.

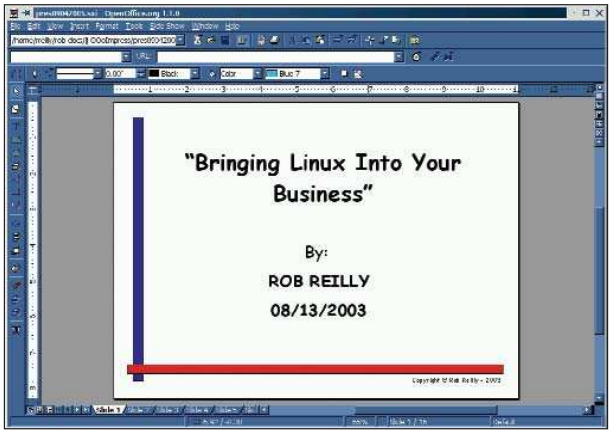
Tegyük fel, hogy nemrég kiválasztottak, hogy Linux-guruként előadást tarts egy jelentős ipari tanácskozáson. A gyomrod föl-leliftezik a felismeréstől, hogy esetleg meg kell tanulnod a Microsoft PowerPoint használatát. Micsoda megaláztatás! Aggodalomra semmi ok, az OpenOffice.org Impress fut Linux alatt. Gondolj csak bele, milyen nyugodtan állsz majd az emberek előtt, tudva, hogy megbízható, régi barátod, a Linux készenlétben áll a vezeték nélküli egér másik végén.

Fő a szórakoztatás

A szakmai bemutatók lényege az ismeretanyag közönségnek történő átadása. A tartalom a műsor, mi vagyunk a műsorvezetők, és az előadás eszközei a színpadi kellékek. A laptopok, rajzok, vetítők, fények és a bemutatókészítő program mind azt a célt szolgálja, hogy összehangolja a látványt, és a hallgatóság figyelmét a lényegre összpontosítsák. Magam is műszaki beállítottságú lévén tudom, hogy a technika sziréne arra csábít, hogy fedezzük fel és babráljunk vele. Ezúttal viszont kénytelenek leszünk ellenállni annak a kísértésnek, hogy állítgassunk: össze kell hoznunk az előadást. Ismertetem tehát az OpenOffice.org Impress segítségével történő diavetítés és -készítés alapjait, szó lesz néhány módszerről, hogy előadásukat miként rendszerezhetjük, majd ráadásként néhány megbízható tanácsot adok, amitől profinak tűnhetünk majd a rajongó közönség előtt.

Mi is az Impress?

Az OpenOffice.org Impress programcsomag segítségével gyorsan szerkeszthetünk és tarthatunk elektronikus diabemutatókat. Használatával beszámolónkhoz ábrákat adhatunk hozzá, emlékeztetőt készíthetünk, továbbá diáinkat weboldalakká alakíthatjuk. Minden feladatot ellát, ami egy kiváló előadás megtartásához szükséges. Mi egyszerűen csak a tartalmat és az egyéni stílust tesszük hozzá.
Az OpenOffice.org Impress betöltése egyszerű. Amennyiben az OpenOffice.org még nincs telepítve, látogassunk el a ➲ http://www.openoffice.org címre, és töltsük le a legfrissebb változatot. Az 1.1-es körülbelül 77 MB-ot tesz ki. Helyezzük a *.gz* fájlt a saját könyvtárba és csomagoljuk ki. Ezután futtassuk a telepítőfájlt, töltsük ki az űrlapot, és készen is vagyunk.
Az Impress használatához nincs szükség 3 GHz-es laptopra.

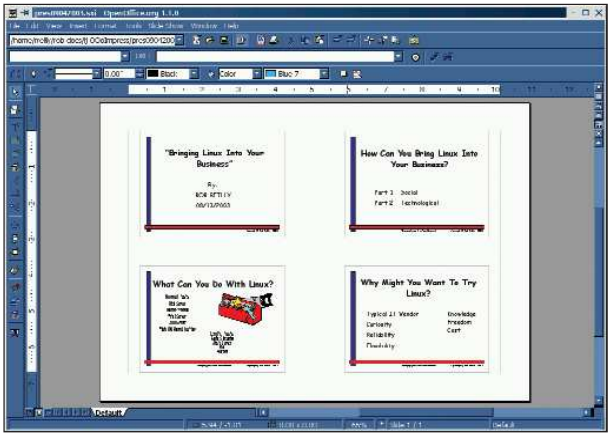


1. kép A bemutató létrehozása az első diánál kezdődik. Ez az Impress fő képernyője

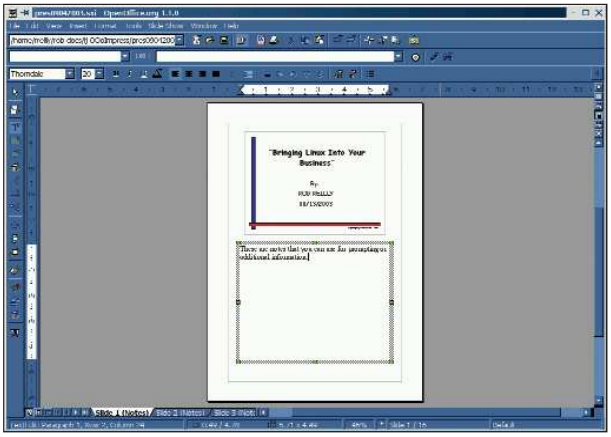
Nagyszerűen működik a régi 166 MHz-es Pentium környezetben is, 8 MB-os videokártyával és KDE vagy bármilyen X ablakkezelővel.

Gyors bemutatókészítés

Izzítsuk be az OpenOffice.org Impresst és tegyünk egy felfedezőutat a programban. Először nézzünk néhány alapszolgáltatást, utána térjünk vissza és rendszerezzük az anyagunkat. Az új bemutatót a legegyszerűbb a ***Tündér*** szolgáltatással indítani. Ez egy egyszerű kezdő vázat ad. Miután bevittük a tartalmat, ábrákkal, szöveggel, animációkkal és formázással egészíthetjük ki.
A bemutató létrehozásához kattintsunk a ***Fájl***/***Új***/***Bemutató*** menüpontra. A következő ablakban válasszuk az ***Üres bemutató*** lehetőséget és nyomjuk meg a ***Tovább*** gombot. Ebben az ablakban a hátteret választhatjuk ki, majd ismét kattintsunk a ***Tovább***-ra. A következő ablakban a diaátmeneteket állíthatjuk be. Első bemutatónkban nincsenek átmenethatások, hanem kézzel kell irányítani a PAGE UP/PAGE DOWN billentyűk vagy az egér segítségével. Tehát kattintsunk a ***Kész*** gombra, mire megjelenik a ***Dia módosítása*** ablak, amelyben meghatározhatjuk a dia elrendezését. Első példánkhoz a „Cím, diagram, szöveg” elnevezésű stílust alkalmazzuk. Jelöljük ki ezt az elrendezést, nyomjuk meg az ***Ok***-t, és nézzük, ahogy első diánk megjelenik (*1. kép*).



2. kép Emlékeztető nézet – oldalanként négy dia



3. kép Jegyzet hozzáadása a Jegyzet nézet felületen

Ezen az első dián a megfelelő helyre beírhatjuk a dia címét, a bal oldalra beilleszthetünk egy képet, jobbra pedig a vázlatpontokat. Az alapértelmezett betűmérettel a vázlatdobozban csak 4–5 sor fér el. Ez azért jó, mert sok tapasztalt előadó egy sereg adatot sűrít egy diára. A diákat és a vázlatot csak súgóként használjuk, mondanivalónkat a saját szavainkkal adjuk elő! Így a hallgatóság olyan szakértőnek tart majd bennünket, aki alaposan ismeri a témát.

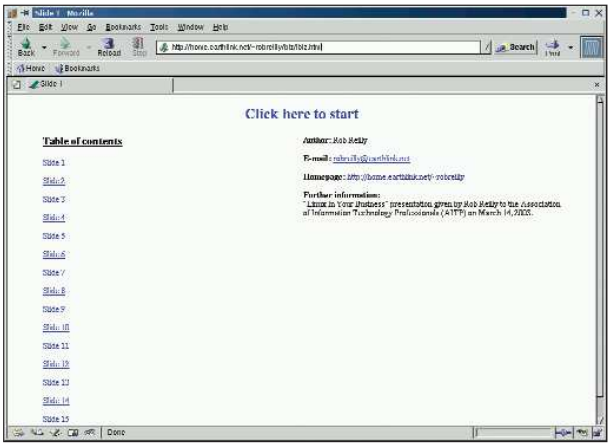
Az első dia létrehozásának feladata 5–10 percet vesz igénybe. Szándékosan mentem végig az egész folyamaton, mert azt akartam, hogy eléggé elmélyedjünk az Impress programban. Azzal, hogy képesek vagyunk egy egyszerű diát létrehozni, birtokába kerülünk annak a tudásnak, ami beszédünk rendszerezéséhez és további diák létrehozásához szükséges.

Felejtsük el, hogy papíron vázoljuk fel a beszámolónkat! Használjuk az Impresst! Ötleteinket gyűjtsük össze és építsük fel logikusan. A címeket fő témakörökként használjuk olyan módon, hogy minden téma egy-egy új diára kerüljön, és 3–4 vázlatponttal részletezzük azokat. Később, amikor a teljes anyag körvonalazódik, átrendezhetjük és újraszerkeszthetjük őket.

Dia, szöveg és ábra beillesztése

Diát könnyű hozzáadni. Először kattintsunk a ***Beszúrás/Dia*** menüpontra. Válasszuk ki a dia típusát, majd nyomjuk meg az ***Ok***-t. A dia megjelenik a képernyőn, és írhatunk bele szöveget, vagy hozzátehetünk bármit, ami tetszik. Ha ezt a diát meg szeretnénk kettőzni, kattintsunk a ***Beszúrás/Dia megkettőzése*** menüpontra. Most már hozzáadhatunk valamilyen szöveget is. Az Impress fő képernyőjének bal felső szélén kattintsunk a ***Szöveg*** ikonra. Mozgassuk a mutatót a dia fölé és valahol nyomjuk meg a bal gombot. Írjunk be egy szöveget. Ezt át is helyezhetjük máshová, ha a szövegdoboz közepénél fogva elhúzzuk. Átméretezéshez jelöljük ki a szöveget és az ablak felső részén található betűméret dobozban adjunk meg egy új méretet.

Képet ugyanilyen könnyű beszúrni, kattintsunk a ***Beszúrás/Kép*** menüpontra, és válasszuk ki azt a képfájlt, amelyet használni szeretnénk. Kattintsunk az ***Ok***-ra, és a kép a diára kerül. Úgy méretezhetjük át, hogy megfogjuk az egyik oldalát, és a képernyő széle felé húzzuk. Ahhoz, hogy az ábrát mozgassuk, középen kell megfogni és a kívánt helyre áthelyezni.



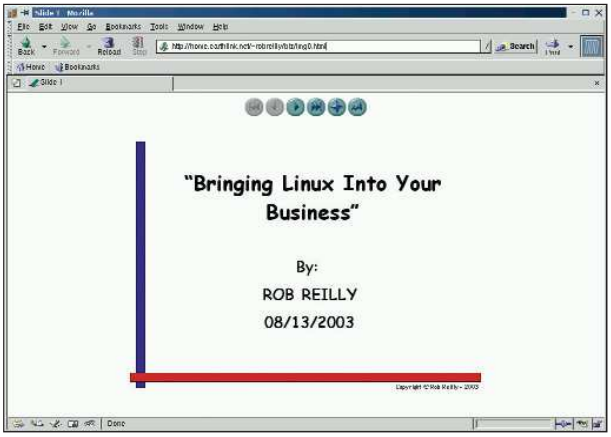
4. kép A főoldal egyéb fontos adatokra mutató hivatkozásokkal

Emlékeztetők és jegyzetek készítése

Az emlékeztető egy füzet alakú diagyűjtemény, amelyben alapértelmezésben négy képkocka szerepel egy oldalon (*2. kép*). Kinyomtatva össze lehet tűzni vagy iratgyűjtőbe lehet tenni, és az előadás kezdete előtt szétosztható a hallgatóság tagjai között. Szakmai előadásokon az emlékeztető segítségével a résztvevők az éppen tárgyalt dia másolatára jegyzetelhetnek.

Emlékeztető készítéséhez váltsunk át emlékeztető nézetre, azaz a jobb felső görgetősáv fölött kattintsunk az ***Emlékeztető nézet*** ikonra. Az emlékeztető nyomtatásához válasszuk ki a ***Fájl/Nyomtatás*** menüpontot, majd nyomjuk meg az ***Ok***-t. A kinyomtatott bemutató minden oldalára négy dia kerül.

A bemutatójegyzetek olyan adatokat tartalmaznak, amelyeket mi és a közönség is hasznosnak találhatunk, de nem láthatóak diavetítés közben. Ha a beszámolónk hosszú és szükségünk van még egy kis segítségre, nyomtassuk ki a jegyzeteket. Jegyzet hozzáfűzéséhez kattintsunk a jobb felső görgetősáv mentén található ***Jegyzet nézet*** ikonra és váltsunk jegyzet nézetre. Válasszuk ki az alsó területet, amiben a „Jegyzet hozzáadásához kattintson ide” felirat látható, és írjunk be egy tetszőleges szöveget. A dia ekkor a képernyő felső felén helyezkedik el (*3. kép*). Végül kattintsunk a Dia nézet ikonra (a ***Jegyzet nézet*** ikon fölött), hogy újra a diát lássuk, jegyzetek nélkül.



5. kép A webes változat előnézete a böngészőben

A bemutató lejátszása

Miután elkészült a remekmű, természetesen próbát szeretnénk tartani – az előadást mindig gyakorolni kell a nagy esemény előtt. Váltsunk teljes képernyős nézetre, és a bemutatót a PAGE UP/PAGE DOWN billentyűk vagy az egerünk segítségével lapozzuk végig. Kattintsunk a ***Diavetítés indítása*** gombra a görgetősáv jobb oldalán (a ***Jegyzet nézet*** ikon alatt), hogy a diavetítés nézetbe lépjünk. Nyomjuk meg az ESC billentyűt, ha az Impress rajznézetéhez szeretnénk visszatérni.

A diák weboldallá alakítása

Eddig a bemutató elkészítésére és előadására összpontosítottunk. De miért ne tennénk elérhetővé a közönség számára a diákat egy weboldalon (hogy később áttekinthessék), hogy igazán hozzáértő előadónak tűnjünk? Ha a hallgatóságot a honlapunkra irányítjuk, egyúttal bemutathatjuk a vállalatunkat, megmutathatjuk az önéletrajzunkat, egyéb bemutatóinkat és kiadványainkat, cikkeinket. Semmi sem tehet egy előadót hitelesebbé, mint az, ha értékes tájékoztatást és szolgáltatást nyújt a hallgatóságának, még akkor is, miután hazamentek. Az a kis külön erőfeszítés, amit azért teszünk, hogy az előadásainkat honlapunkon elérhetővé tegyük, meg fog különböztetni bennünket a kispályás előadóktól.
Az OpenOffice.org Impress megkönnyíti azt a folyamatot, hogy diáinkból egyszerű weboldalt készítsünk. Nyissuk meg a bemutatónkat az Impressben és kattintsunk a ***Fájl/Exportálás*** menüpontra. Jelöljük ki a munkakönyvtárat a HTML és a képfájlok számára. Ezután írjuk be a főoldal fájlnevét (a ***.html*** kiterjesztés nélkül) és kattintsunk az ***Exportálás***-ra. A ***Látványterv hozzárendelése*** képernyőn kattintsunk a ***Tovább*** gombra. A ***Kiadványtípus*** ablaknál is a ***Tovább***-ot nyomjuk meg. A következő képernyőn a képeket JPEG formátumban menthetjük, tehát válasszuk a ***Közepes felbontás***-t, 800×600 képpontot. Töltsük ki a szerző, e-mail cím, honlap és további információk rovatokat az ***Információk a címlaphoz*** képernyőn. Ezután jelöljük meg ***Az eredeti bemutató*** másolatára mutató hivatkozás jelölőnégyzetet és lépjünk tovább. Válasszuk ki a böngésző-gombok megjelenését, és kattintsunk a ***Tovább*** gombra.
A ***Színsémák kiválasztása*** ablakban válasszuk a ***Létrehozás***-t. Végül a ***HTML-terv neve*** ablakban menthetjük a tervet, amit alkottunk. Tegyünk így, és készen is vagyunk.

Most már van egy kitűnő weblapunk, ami keretezi a diáinkat. A címoldalon látható a nevünk, a levélcímünk, a honlapunk címe és a további adatok (*4. kép*), valamint egy hivatkozás az eredeti Impress bemutatónkra.
Ahhoz, hogy kezünk munkáját megcsodálhassuk, menjünk a munkakönyvtárba, és nyissuk meg a HTML-fájlt, amit az első exportáláskor hoztunk létre (lásd az *5. képet*).
Amennyiben a weboldal kinézetével elégedettek vagyunk, az egész könyvtárat töltsük föl a honlapunkra. Ne feledkezzünk meg az eredeti vagy az ***.sxi*** fájlról sem. Egy részletes hivatkozás mutasson rá a honlapunkról, és adjuk meg a címet a hallgatóságnak. Feltöltés után ellenőrizzük a bemutató oldalait, hogy meggyőződjünk róla, hogy minden ott van-e és működőképes.

Reflektorfényben

Két beszámoló előtti feladat is akad, amit a későbbek folyamán érdemes megtenni ahhoz, hogy az előadás sikerét bebiztosítsuk. Azért beszélsz majd annyi ember előtt, mert a téma szakértője vagy. Mégis lazíts és érezd jól magad a házigazda szerepében, tedd élvezetessé az előadásodat.
Másodszor az előadást biztató szavakkal zárd, mondjuk valami ilyesmivel: „Javaslom, hogy amikor elmennek innen, az első teendőjük az Impress kipróbálása legyen!”, ahelyett, hogy egyszerűen köszönetet mondanál, és mosolyogva fogadnád a mennydörgő tapsvihart.
Hacsak nem vagy rendkívül tapasztalt szónok, legalább kétszer teljes egészében próbáld el az előadást, mielőtt élőben szerepelnél. A próbát úgy értem, hogy szerezz egy termet, kösd össze a laptopot és a vetítőt és valós időben vedd át az egész előadást. Ne félj stoppert használni, hiszen teljesen biztosnak kell lenned abban, hogy az előadás belefér a megszabott időbe. Valószínűleg szükség lesz még egy kis szerkesztésre. A legjobb, ha már akkor elkezdjük összeállítani a bemutatót, amikor megkapjuk a feladatot. Semmi nem ronthat el jobban egy előadást, mint amikor kifutunk az időből vagy nem fedjük le az egész témát. Minden eshetőségre fel kell készülni. Ha például az előző szónok túl sokáig beszélt, mi a teendő?
Az előadás előtt próbáld ki helyben, hogy működik-e a laptopod a vetítővel. Végül vigyél magaddal mindent, amire szükséged lehet, beleértve a fóliákat és a hosszabbítókat (a biztonság kedvéért).

Összegzés

A szakmai bemutatók hasznosak és szórakoztatóak is lehetnek. Használd az Impresst a hordozható gépeden, és élvezd a reflektorfényt. Tervezz előre, mindenképp próbálj eleget, és ne csússz ki az időből.
Sok szerencsét!

Linux Journal 2004. március, 119. szám

Rob Reilly (robreilly@earthlink.net)
Szakíró és -előadó, akinek a cikkei a LinuxToday.com-on és a PC Update magazinban jelennek meg.

A magyar Linux: UHU (2. rész)

Folytatjuk az előző hónapban megkezdett barangolásunkat a magyar fejlesztésű UHU Linux-rendszerben.

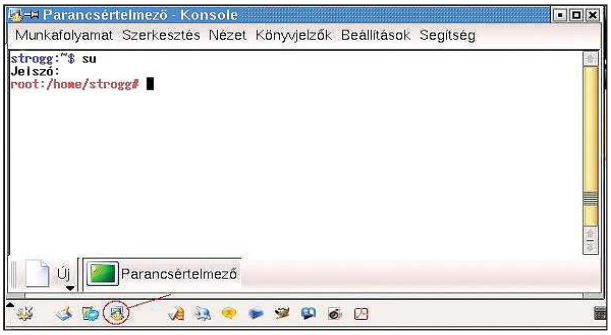
Sorozatunknak ebben a részében néhány fontos alapszolgáltatást veszünk közelebbről szemügyre, ez reményeim szerint jelentősen megkönnyíti azok számára az első próbálkozásokat, akik csak most ismerkednek ezzel a remek Linux-változattal.

Az UHU-ban használatos alapvető parancsok

Mint minden Linux-változat, ez a terjesztés is remekül vezérelhető parancssorból. Erre két fő lehetőség is rendelkezésünkre áll: az egyik, hogy a már futó ablakkezelőben nyitunk egy konzolt.
Az *1. képen* jól látható, honnan is indítottuk a konzolt.
A kis karikában lévő hivatkozásra kattintva a KDE-felületen, közvetlenül a rendszer vezérléséhez kapunk felületet.
A képen látható konzolban szintén pontosan nyomon követhetjük, hogyan is veszi át a vezérlést a rendszergazda, bármilyen felhasználói névvel lépjünk is be. Itt lehetőségünk nyílik rendszergazdaként bejelentkezni. Ehhez az 1. képen is szerepelő `su` parancsot használjuk:

```
su
password: <adjuk meg a rendszergazdai jelszót>
root:/pillanatnyi könyvtár/ <rendszergazdaként
látjuk a pillanatnyi könyvtárat>
```

Ettől a pillanattól kezdve immár rendszergazdaként dolgozhatunk a rendszerben. Figyelem, a rendszergazdának mindenhez van jogosultsága és bármit megtehet a rendszerben! Tehát kényes műveletek végrehajtására is lehetősége nyílik, s egy hibás utasítás a teljes rendszerre ugyanilyen kritikus hatással lehet.
Továbbá soha, de soha ne próbáljunk rendszergazdaként elindítani internetes böngészőt, IRC-ügyfelet vagy levelező-ügyfelet. Ilyenkor bárminek, ami az internetről bejöhet, olyan hatása lehet, mintha az adott internetről származó adatot a rendszergazda engedélyezte volna, ez pedig komoly támadási felületet kínál.
Minderre azért lehet szükség, mivel számos parancs megkövetelheti, hogy csak és kizárólag a rendszergazda használja. Több olyan parancs is létezik, amit csak a rendszergazda adhat ki, mivel módosítja a rendszerbeállításokat, esetleg olyan rendszerfájllal, könyvtárral dolgozik, amelyhez csak a rendszergazdának van jogosultsága. A másik



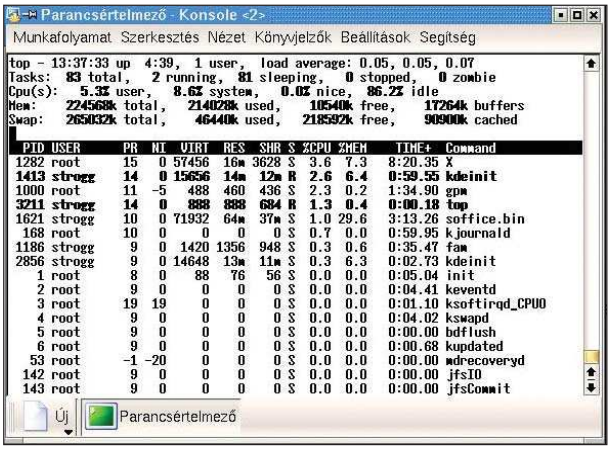
1. kép A már futó ablakkezelőben egy konzolt nyitunk

ilyen megoldás, ha egy teljes képernyőt adunk ennek a felügyeleti felületnek. Ehhez csak a CTRL+ALT+F2....6 billentyűkombinációkat kell lenyomni, és pontosan ugyanolyan felületet kapunk teljes képernyőn, mint amit az előbb ablakban láttunk. A különbség mindössze annyi, hogy mindezt DOS-ra emlékeztető stílusban kapjuk meg. Ennek a megoldásnak is megvan a haszna, mint azt később is látni fogjuk (például ezzel a módszerrel F2...6-ig több ilyen konzolt is nyithatunk, és mindegyikre más felhasználóként vagy akár rendszergazdaként bejelentkezhetünk; a kiindulási grafikus felületre a CTRL+ALT+F7 kombinációval lehet visszatérni). Egyetlen veszélye a dolognak, hogy a rendszergazda a visszalépés előtt nem jelentkezik ki (`exit` parancs), ezáltal továbbra is a rendszerben fog tartózkodni. Ekkor, ha valaki a gépe elé ül, a fenti billentyűparancsot leütve korlátlanul garázdálkodhat a rendszerben.
Ezt azért említem meg, mert igen gyakori felhasználói hiba, és rendkívül kényes helyzetet teremt a rendszer biztonsága szempontjából. A rendszer gyakorlatilag mindenki előtt tárva-nyitva áll: rögtön teljes uralmat biztosít a gép fölött annak, aki eléül, ezért munkánk végeztével sose feledjünk el minden rendszergazdaként használt konzolból vagy programból kilépni – addig semmilyen körülmények között ne hagyjuk magára a gépünket!
A rendszergazdaként és a felhasználóként használatos alapvető parancsokat *táblázatunkban* mutatjuk be (➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/UHU/tablazat.html) részletesebben. A bal oldalon láthatjuk a parancsot, azt jobbra haladva az alapvető írásmód követi,

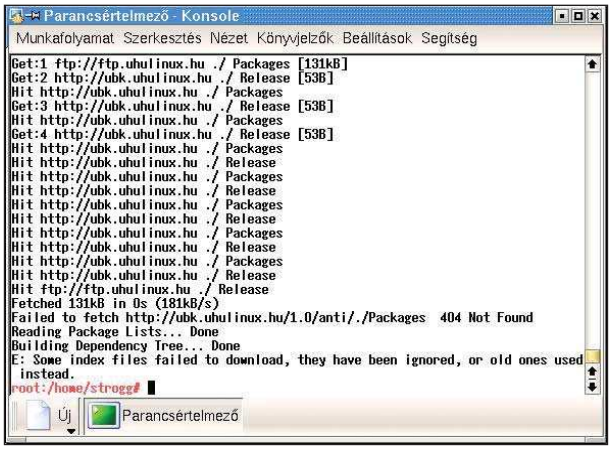
majd a segítségkérés módját és az alkalmazásához szükséges jogosultságot olvashatjuk.
A jogosultságoknál a „mindenki” egész pontosan azt jelenti, hogy habár a parancsot a felhasználó ki tudja adni, ha olyan fájlhoz vagy szolgáltatáshoz akar vele hozzáférni, ami a rendszergazda hatáskörébe tartozik, akkor hibaüzenetet fog kapni. A „rendszergazda” megjelölés pedig azt jelenti, hogy az adott parancs kizárólag a rendszergazda számára érhető el, a felhasználó nem tehet vele semmit (*2. kép*).

A rendszer frissítése

Meglévő rendszerünket rendszeresen frissíteni tudjuk, ha akarjuk. Olykor bizony szükséges is, hiszen ha biztonsági rést találnak valamelyik programban, annak a frissítését célszerű letölteni és telepíteni. Erre két mód is adódik: egyrészt a már meglévő csomagokat, ha CD-n kapjuk meg



2. kép Példa a top parancs kimenetére



3. kép Az apt-get működés közben

(több lelkes felhasználó felírja CD-re, csak keresni kell), kézzel a KPackage programmal (használatáról egy kicsit lejjebb szót ejtünk) telepíthetjük; másrészt az `apt-get` parancsot használhatjuk. Ez a programocska tökéletesen elvégzi a feladatát, egyetlen előfeltétele van: üzembiztos és állandó internetkapcsolat. Saját időnk érdekében nem árt, ha ez a kapcsolat széles sávú, gyors elérés. Ezt a programot kizárólag a rendszergazda használhatja, tehát nyissunk meg egy konzolt és jelentkezzünk be `su`-val. Ekkor több lehetőségünk is adódik. Mielőtt azonban munkához látnánk, még az első indítása előtt célszerű elvégezni a következőket: az `apt-get` a */etc/apt/sources.list* fájlból veszi azokat a kiszolgálóadatokat, amelyeken keresni fogja a csomagokat. Tehát adjuk ki a `kwrite` parancsot, és mivel rendszergazdaként vagyunk bejelentkezve, az így elinduló kis szövegszerkesztő minden fájlt képes lesz felülírni, például a *sources.list*-et is. Ezután egy egyszerű böngészővel (mondjuk a Konquerorral) menjünk fel a ➲ http://ubk.uhulinux.hu weboldalra. Az itt található hivatkozások gyűjteményét – ami a csomagokat készítő lelkes felhasználók weboldalaira mutatnak – másoljuk be ebbe a *sources.list*-be. Vigyázzunk rá, hogy egyetlen sora se kezdődjön szóközzel – ugyanis biztosan találunk több ilyet is –, tehát a sorok elől vegyük ki a szóközöket, hogy közvetlenül a `deb` szócskával kezdődjenek. Ezután mentsük a *sources.list* fájlt és lépjünk a ki a `kwrite`-ból. Majd rögtön tudassuk az `apt-get` programmal, hogy több kiszolgálóról is dolgozhat. Ezt a következő paranccsal tehetjük meg:

```
apt-get update
```

Futás közben jól láthatjuk, hogy az összes, a *sources.list*-ben szereplő kiszolgálót megvizsgálja és a tartalmáról listát készít magának. Ekkor teljesen készen állunk, hogy bármi kedvünkre valót telepítsünk vagy csak frissítsük a rendszerünket (*3. kép*). Ilyen módon akár több feladatot is elvégezhetünk, ezenkívül egy-két dolgot az `apt-get` önmagától is végrehajt. Például képes arra, hogyha egy adott csomag működéséhez egy másikra van szükség, azt is letöltse, de frissíti is az adott csomagot, ha ez lenne a telepítendő csomag függőségi követelménye. Alapértelmezettként leellenőrzi, hogy a rendszerünkben lévő csomagok változatszáma megegyezik-e a listájában szereplő friss csomagok változatával, és kiírja, hogy hány olyan csomagot talált, amelyekből a kiszolgálókon frissebb is csücsül. Ha utasítjuk erre, rögtön hozzá is lát és letölti a frissebb csomagokat, majd azonnal telepíti is őket. Ezt a következő paranccsal tehetjük meg:

```
apt-get upgrade
```

Ha csak a teljes rendszerünket akarjuk frissíteni, akkor a következőt használhatjuk:

```
apt-get dist-upgrade
```

Amennyiben új csomagot szeretnénk telepíteni, az alábbi utasítást írjuk be:

```
apt-get install csomagnév
```

Egy telepített csomagot pedig ilyen módon távolíthatunk el:

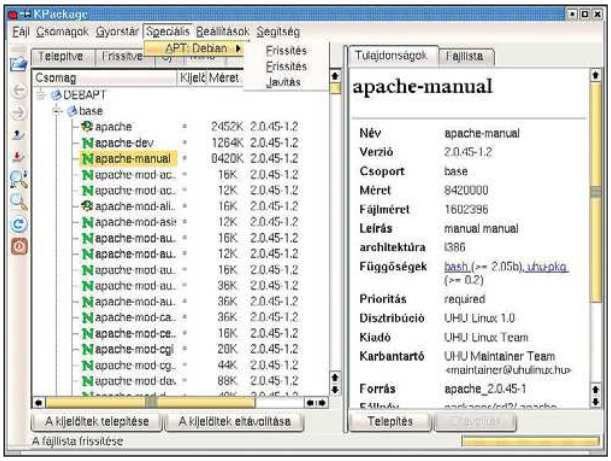
```
apt-get remove csomagnév
```

Hasznos dolog megjegyezni, hogyha esetleg nem tudnánk a csomag nevét, akkor sincs semmi baj, hiszen ilyenkor a

parancs után elég csak a csomag feltételezett kezdőbetűjét odaírni, majd lenyomni a TAB billentyűt. Ekkor a konzolhoz hasonlóan, önműködően kiírja a listájában szereplő összes ilyen betűvel kezdődő csomag nevét.
Ezután a program megnézi, hogy a csomag létezik-e, s ha igen, akkor mi a változatszáma (ha már fel lett telepítve, szól, hogy már fenn van és ez a legfrissebb változat), valamint a függőségeket is megnézi, sőt felajánlja, hogy azokat is letölti nekünk. Amikor elkészült a letöltésekkel, önműködően hozzá is lát a felpakolásukhoz. Természetesen olyan módon is kiadhatjuk a parancsot, hogy a csomagot csak letöltse, de ne telepítse. Az összes létező kapcsolóról a `help` vagy a `man apt-get` utasítás ad részletesebb felvilágosítást.

A KPackage csomagkezelő

Az UHU Linux alapértelmezett csomagkezelője a KPackage. Ezzel a grafikus programmal tudunk ***.uhu***-csomagokat telepíteni, a felrakott csomagok között keresni, törölni, egyszóval mindent megtenni, amit csak lehetséges. Nagyon hasznos tulajdonsága, hogy rengeteg adatot ír ki az adott csomagról, például a készítőt, az elérhetőségét, a honlap címét, a függőségeket, a telepítési útvonalakat, akár a csomagban található fájlok listáját is lekérhetjük tőle (*4. kép*).



4. kép Az UHU Linux alapértelmezett csomagkezelője a Kpackage

Fontos, hogy az összes, csomagokkal kapcsolatos művelethez meg kell adni a rendszergazda jelszavát. Igaz, a program futása során elég csak egyszer megtenni mindezt. Ez a program ügyel arra is, hogy olyan csomag ne kerülhessen be a rendszerbe, amely ütközést okoz vagy a függőségek miatt működésképtelen lesz. A telepítési párbeszédpanelen ezt természetesen felülbírálhatjuk (kivesszük a pipát a függőségek vizsgálata elől), de ekkor sérülhet a rendszer, az `apt-get` pedig biztosan. Ilyenkor az `apt-get` addig nem hajlandó rendesen működni, amíg a hibát ki nem javítottuk. Ezt a Kpackage programban a 4. képen is látható ***Speciális/APT: Debian/javítás*** menüpontja alól tehetjük meg. (Ez a szolgáltatás az `apt-get` bármely hibája esetén gyógyulást hoz neki.)

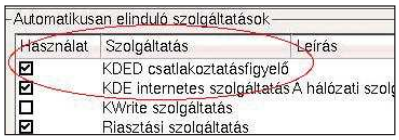
Az nVidia videokártyák kérdésköre

Egy kezdő számára minduntalan felmerül a kérdés, hogy hogyan és mivel lehetséges az UHU 1.x rendszerre úgy telepíteni az nVidia videokártyát, hogy annak a 3D-szolgáltatása is elérhető legyen. Ennek ugyanis – mint hamarosan látni fogjuk – jó hasznát vesszük, ha játszani szeretnénk.
Az nVidia szerencsére sokkal komolyabban veszi a linuxos felhasználókat, mint a legtöbb gép- és alkatrészgyártó.
Ez hosszú távon fenn fogja tartani a piaci előnyét a versenytárs gyártókkal szemben, ezért mindig számíthatunk arra, hogy a meghajtók frissítése nem kerül takaréklángra.
A meghajtó telepítése gyerekjáték, mindössze pár műveletet kell elvégeznünk. Töltsük le a ➲ http://www.nvidia.com oldalról az éppen időszerű meghajtócsomagot. Látni fogjuk, hogy ennek a fájlnak a kiterjesztése ***.run***, ez futtatható konzolban. Erre azért van szükség, mert az nVidia nem teljesen nyílt forrású meghajtót bocsát a felhasználók rendelkezésére, inkább vállalja, hogy ő adja ki őket. Ezért csak egyetlen referenciameghajtó létezik a Linuxban, ami semmi másra nem alkalmas, minthogy a kívánt felbontásban meg tudja jeleníteni a képet a kártyánkon. Ha akár grafikusan, akár konzolból letöltöttük a fájlt, tegyük futtathatóvá. Ezután lépjünk ki a grafikus felületből. Nyomjuk le a CTRL+ALT+F2 billentyűkombinációt, és ezen a konzolon rendszergazdaként jelentkezzünk be. Ha ez megvan, indítsuk el az imént letöltött ***.run*** fájlt. Az nVidia által javasolt felhasználási szerződés elfogadása után megvizsgálja a rendszermagot, és ha nem találja megfelelőnek vagy gondja van vele, akkor felkínálja, hogy az FTP-kiszolgálóról letölti a megfelelőnek ítélt alkotóelemeket. Ha ez felmerülne (szinte biztosan ez fog történni), akkor engedélyezzük neki.
Ezután nekiáll és közvetlenül lefordítja a megfelelő modulokat számunkra. Fontos tudni, hogy az nVidia-meghajtók mindig két fő alkotóelemből állnak: az egyik a rendszermaghoz való modul, amely a kártya használatát adja meg a rendszernek. A másik összetevő a `GLX` modul, amely a 3D-s megjelenítésért felel. Egy 3D-s játékhoz mindkettőre szükség van. Az *6. képen* éppen a magmodul létrehozása és telepítése látható.
Ha ezzel elkészültünk, még semmi sincs rendben, ugyanis hátravan még az UHU grafikus felületért felelő fájljának a szerkesztése. Célszerű, ha rögtön a telepítés után

5. kép A run megvizsgálja a rendszermagot

6. kép A magmodul létrehozása és telepítése

7. kép XF86Config fájl szerkeszése



8. kép A KDED csatlakoztatásfigyelő

meghívjuk (még mindig rendszergazdaként) a Midnight Commandert (az `mc` parancs segítségével), és ott a megfelelő fájlra navigálva az F4-gyel megnyitjuk szerkesztésre. Az elején keressük ki a `load glx` sort. Amennyiben e sor előtt nincs egy kettős kereszt (#), akkor semmi dolgunk nincs vele, minden rendben van. Ha mégis ott lenne, vegyük ki belőle. Ezt követően keressük ki a `Section "Device"` sort, amely a parancsfájl első egyharmadában található, s ebben keressük meg a `Driver "nv"` sort. Ha szeretnénk a más rendszerben megszokott árnyékoltkurzor-szolgáltatást is használni, tegyük alá az `Option "CursorShadow"` sort is. (Figyelem, a szolgáltatás legalább GeForce vagy jobb kártya esetén működik!) A `Driver "nv"` sort vagy a *7. képen* látható módon, vagy pedig a következőre javítsuk át:

```
Driver "nvidia"
Option "CursorShadow"
```

Ha ezzel is megvagyunk, akkor mentsük a fájlt, és indítsuk el (vagy újra) az X grafikus felületet. A sikerről úgy tudunk meggyőződni, hogy az X indulásakor egy teljes képernyős, fehér alapú nVidia logót fogunk látni. Ekkor minden rendben zajlott és a művelet sikeres volt, az nVidia kártya telepítve van, új rendszerünk immáron játékra is teljesen alkalmas. Akinek egyébként részletes leírásra van szüksége az nVidia kártyák Linux alatti beállításáról, az egy teljes angol nyelvű leírást talál ***.pdf*** formátumban az nVidia honlapján vagy az FTP-kiszolgálóján.

Az Unreal Tournament telepítése

Erőfeszítéseinket megkoronázandó lazítsunk egy kicsit a méltán népszerű Unreal Tournament nevű játékkal. Mindenekelőtt két dologra lesz szükségünk hozzá: a játék telepítőlemezére, valamint egy linuxos telepítőre. A telepítőt a
➲ http://www.lokigames.com oldalról vagy a
➲ http://www.icculus.org/~ravage honlapról is letölthetjük. Az utóbbin nemcsak ennek a játéknak a telepítője érhető el, hanem például a teljes Unreal-sorozaté is. Ha mindkettő megvan – a Játék rovatban a 80. oldalon bővebben is olvashatnak róla –, akkor már csak egyetlen nehézséget kell megoldani. Mégpedig azt, hogy amikor ennek a játéknak a telepítője elkészült, még nem létezett az `automount` nevű önbefűző eljárás. (Ez hivatott gondoskodni arról, hogy elég egy lemezt betenni a meghajtóba, az önműködően befűződik a rendszerbe, ellentétben a régebbi, kézzel történő rendszerbe fűzéssel.)
Sajnos így hiába tesszük be a telepítő CD-t, a program állandóan kéri majd, mivel sehogyan sem fogja észrevenni. A megoldás pedig elég egyszerű: mielőtt behelyeznénk a lemezt, indítsuk el a ***KDE vezérlőpult***-ot. Itt keressük ki a ***KDE komponensek/szolgáltatások*** részt. Ekkor a jobb oldalon látni fogjuk, hogy létezik egy ***KDED csatlakoztatásfigyelő*** sor, amelynek állapota: ***fut*** (*8. kép*).
Ezt a ***leállítás*** gombbal állítsuk ***nem fut*** állapotba, ezt követően kattintsunk az ***alkalmazás*** gombra. Ha ez megvan, nyissunk ismét konzolt és jelentkezzünk be rendszergazdaként. Hozzunk létre egy befűzési pontot a CD-nek a következő módon, azután nézzük meg a `cat` paranccsal, hogy mi a CD pontos elérési útja.

```
mkdir /mnt/cdrom <Befűzési pont létrehozása.>
cat /etc/fstab <Az fstab állományból lekérjük
a bejegyzett meghajtókat.>
```

Ezután helyezzük be a CD-t, és rögtön adjuk ki a befűzési parancsot az ***fstab***-ból vett CD elérési útjának adatai alapján, nagyjából hasonlóan:

```
mount /dev/ide/host0/bus0/target0/lun0/cd  /mnt/cdrom
```

Ha ekkor felzúg a meghajtó, sikerrel jártunk, és bátran elindíthatjuk a telepítőprogramot. Fontos, hogy ezt is rendszergazdai módban tegyük meg, mivel ez a ***/usr/local/games*** könyvtárba fog írni, ehhez pedig csak a rendszergazdának van jogosultsága (*9. kép*).

Végszó

Ebben a hónapban átnéztük, milyen alapvető parancsokat célszerű elsajátítani; a számuk nem is sok, és az első lépések megtételéhez bőven elegendőek. Ezzel jó néhány kérdést előre megválaszoltunk. Beszéltünk a rendszer frissítéséről, a csomagkezelésről, annak tulajdonságairól és két remek programot ismertettünk, amelyekkel ezeket a feladatokat hiba nélkül elvégezhetjük. Telepítettünk egy nVidia gyártmányú videokártyát, és reményeim szerint sokakat átsegítettünk az Unreal Tournament telepítésének egyik komoly buktatóján. Sorozatunk következő részeiben elmélyedünk az UHU Linux-rendszerben, minden hónapban egy-egy főbb témakört járva körül. Legközelebb a grafikai alkalmazásokat vesszük szemügyre, hiszen ebben a Linux-változatban is akad jó néhány kiváló, témába vágó program.

Sok sikert és remek szórakozást az új rendszerrel, illetve a játékkal!

Dancsok „strogg” Zoltán (strogg@mail.tvnet.hu) Jelenleg technikai szerkesztőként dolgozik a BME-OMIKK-nál, ahol oktat is. Emellett egyetemi képzésben vesz részt, programozó matematikus szakon. Négy éve foglalkozik Linuxszal. Szabadidejében operációs rendszereket gyűjt és weblapot vezet.

Számítógépes hálózatok (4. rész)

Először a fizikai réteggel, illetve az adatátvitel alapjaival ismerkedünk meg.

Az adatátvitel elméleti tárgyalásakor az a gond merül fel, hogy meglehetősen sok matematikai és fizikai kérdésre kellene választ adnunk. Ha nagyon igényesek lennénk, akkor mondandónkat a Fourier-sorokkal kezdenénk, periodikus függvényekkel varázsolgatnánk és némi integrálszámítást is végeznénk. Ez viszont a közérthetőség rovására menne, továbbá e cikk nem lenne népszerű azok körében, akik mélységes megvetéssel viszonyulnak az elemzés szép, ám egy kicsit sajátos gondolkodást igénylő világához. Azt is megtehetnénk, hogy egyből az adatkapcsolati réteggel kezdünk – ez azonban a sorozat teljességét veszélyezteti, hiszen a fizikai réteg minden hálózat alapja.
Ezért arra kényszerülünk, hogy egy-két részlet erejéig belemélyedünk ugyan a fizikai rétegbe és az adatátvitel elméleti, illetve gyakorlati kérdéseibe, de a tüzetesség igénye nélkül. Csak arra tértünk ki, ami igazán fontos (vagy érdekes).

Az adatátvitel elméleti alapjai

Az első kérdés az, hogy miként tud az egyik számítógép a másikkal egy bizonyos csatornán keresztül adatot közölni. Nyilván úgy, hogy a csatorna egy bizonyos fizikai jellemzőjét megváltoztatja. Ha a csatorna egy vezeték, akkor ilyen változtatható fizikai jellemző lehet például a feszültség vagy az áramerősség. Az adatok továbbítására használhatnánk akár elektromágneses hullámokat is, ilyenkor a hullám amplitúdója, esetleg frekvenciája lehet az adat hordozója.
Az adatcsatorna legfontosabb tulajdonsága a sávszélesség. Azt mindenki tudja, hogy minél „szélesebb sávú” az adott vonal, annál gyorsabban tudunk filmeket letölteni az internetről. A sávszélesség azonban nem azt határozza meg, hogy milyen gyorsan tudunk egy bitet átküldeni. Ez az úgynevezett adatsebesség vagy átviteli sebesség. A sávszélesség analóg alapú rendszerekben játszik szerepet, és az adott analóg jel legnagyobb és legkisebb frekvenciájának különbségét értjük alatta.
Analóg rendszerre a legjobb példa az emberi beszéd, ennek alsó frekvenciája 300 Hz, a felső 3300 Hz. Egyszerű matematikai lépéssel kiszámítható, hogy az emberi beszéd sávszélessége 3300 – 300 = 3000 Hz. Talán nem meglepő, hogy a hagyományos telefonhálózatok is ekkora sávszélességgel bírnak.
Az adatcsatorna másik fontos tulajdonsága a jelzési sebesség. Ha erről nem is, de ennek mértékegységéről, a baudról már biztos sokan hallottak. Ez nem más, mint a felhasznált jel értékében 1 másodperc alatt bekövetkező változások száma. Ezt a következő képlet segítségével könnyedén kiszámíthatjuk:

$$1 \text{ baud} = \log_2(P)$$

Ebben a P a jelszintek száma. Sokan a baudot tévesen az adatsebesség szinonimájaként használják. Nekik csak abban az esetben lehet igazuk, ha két jelszinttel dolgozun ha például nyolc különböző feszültségi szintet használnánk, akkor egy szintváltással 3 bitet is továbbíthatnánk. Ez azt jelentené, hogy az adatsebesség a jelzési sebesség háromszorosa lenne.
Egy csatorna legnagyobb adatsebessége szoros kapcsolatban áll a jelzési sebességgel és a sávszélességgel. ***H. Nyquist*** 1942-ben állította fel azt a tételt, amely fényt derít e három mennyiség kapcsolatára:

$$\text{maximális adatsebesség} = 2H \log_2 P$$

Ebből a H a csatorna sávszélessége. Ezt egyből a gyakorlatba átültetve azt mondhatjuk, hogy egy 16 állapotú kódolást használó telefonvonalon (amelynek 3 kHz a sávszélessége, előbb kiszámoltuk) 24 000 bit/sec az elérhető legnagyobb sebesség.
Ha egy kicsit elgondolkozunk, arra a következtetésre juthatunk, hogy a jelzési szintek növelésével tetszőleges átviteli sebességet érhetünk el az adott csatornán. Tapasztalataink azonban azt mutatják, hogy a természet mindenféle fizikai korlátok felállításával mindig megnehezíti a dolgainkat. Így hát elég bizarr világ lenne az, ahol egy teljes DivX-ben kódolt brazil szappanoperát pár másodperc alatt átküldhetnénk a barátunknak a hagyományos telefonhálózaton keresztül.
Az, hogy egy csatorna mennyire „zajos”, azt az úgynevezett jel–zaj-viszony mondja meg. Ha az S a jel teljesítménye, az N pedig a zaj teljesítménye, akkor a jel–zaj-viszonyt az S/N

képlet segítségével kaphatjuk meg. A jel–zaj-viszony megadása azonban egy kicsit trükkös, ugyanis nem az S/N értékét, hanem ennek tízes alapú logaritmusának tízszeresét használjuk, ennek a mértékegysége a decibel (dB). Ha az S/N értéke 10, akkor az 10 dB-t jelent, ha az S/N értéke 100, akkor 20 dB és így tovább.
És valóban: a fenti képletet csak a zajmentes csatornák esetében használhatjuk. A való életben azonban minden csatorna véletlenszerű zajjal terhelt – ez azért jelent korlátot az adatsebességben, mert egy bizonyos jelszint felett képtelenek leszünk megkülönböztetni a jelzési szintet a zajtól. A zajjal terhelt csatornákra vonatkozó legnagyobb adatsebesség kiszámítására ***Shannon*** talált egy képletet, amely a következőképpen néz ki:

$$\text{maximális adatsebesség} = H \log_2(1 f S/N)$$

A hagyományos telefonhálózaton a jel–zaj-viszony körülbelül 30 dB, így az ott elérhető legnagyobb adatsebesség körülbelül 30 000 bit/sec. Ez nem azt jelenti, hogy ekkora sebességgel még tudunk adatokat küldözgetni. Ez a képlet egy felső korlátot ad, amit a gyakorlatban csak nagyon-nagyon ritkán sikerülhet elérni.
Shannon képletének létezik még egy fontos tanulsága: ha nagyon nagy a zaj, akkor is lehetséges az adatátvitel, igaz, elég kis sebességgel. Ezért van az, hogy a több millió kilométerre lévő űrszondákkal is tudunk kapcsolatot tartani, hiába ezerszer gyengébb a jelük, mint a háttérzaj.
Most pedig nézzük meg, milyen átviteli közegekkel is találkozhatunk a gyakorlatban. Először a vezetékesekkel, majd a vezeték nélküliekkel foglalkozunk.

Csavart érpár (twisted pair)

Ez egy régebbi, ám még ma is rendkívül elterjedt átviteli közeg. Két, jellemzően 1 mm vastagságú szigetelt rézhuzalból áll, amelyek spirálszerűen egymás köré vannak tekerve. A sodrásnak az az értelme, hogy ezzel csökken a kettejük közti elektromágneses kölcsönhatás.
Mivel a csavart érpár akár több kilométeres szakaszon is erősítés nélkül használható, a legfontosabb felhasználási területe a távbeszélő rendszerek kiépítése. Az esetek túlnyomó többségében a telefonokat ilyen vezetékek kötik össze a központtal.
A csavart érpárt analóg és digitális jelátvitelre egyaránt használhatjuk. A sávszélesség erősen függ a vastagságtól és az áthidalandó távolságtól. Ha azonban csak 1–2 km-en belül szeretnénk adatokat továbbítani vele, akkor általában néhány Mb/s-os sebességre számíthatunk.
A számítógép-hálózatok terén is fontos szerepet tölt be, habár számos változata közül itt csak kettővel találkozhatunk. A hármas kategóriájúban négy vezetékpárt fogtak össze egy műanyagtokkal, amely egyrészt védi, másrészt egybentartja őket. Amerikában 1988 előtt elterjedt volt, hogy minden irodaház emeletére egy ilyen kábelt vezettek be, amely egy elosztó segítségével az összes irodahelyiségbe elágazott. Így el lehetett érni, hogy minden irodában négy hagyományos telefon vagy két kétvonalas telefon legyen. 1988 után inkább az ötös kategóriájú csavart érpárokat kezdték használni, ami annyiban volt jobb elődjénél, hogy egységnyi hosszon többet csavarodott. Ez nagyobb távolság esetén is jobb minőségű átvitelt eredményezett, így nagyszerűen használhatók voltak a számítógépes hálózatok kiépítéséhez.
Ennek a két kábelfajtának a közismertebb neve az árnyékolatlan csavart érpár (Unshielded Twisted Pair, vagy röviden UTP). A csavart érpár azonban olcsó és ennek köszönhetően várhatóan a jövőben sem fog veszíteni a népszerűségéből.

A „koax"

A másik elég széles körben használt átviteli közeg a koaxiális kábel. Legfontosabb tulajdonsága, hogy jobb árnyékolással rendelkezik, mint a csavart érpár, ezért nagyobb távolságokon gyorsabb átvitelre képes.
A koaxiális kábel közepén egy tömör rézhuzalmag található; ezt egy szintén vastag szigetelő veszi körül. Ezt is körülveszi egy sűrű szövésű vezető anyag, amit műanyagborítás fed a mechanikai sérülések ellen. Ennek a szerkezetnek köszönhetően a koax zajszintje viszonylag kicsi és nagy sávszélességű jelek továbbítására is képes. Természetesen az elérhető sávszélesség nagymértékben függ a kábel hosszától, de egy kb. 1 km hosszú koaxon akár 1–2 Gb/s-os sebességet is könnyen elérhetünk. Hosszabb kábelek alkalmazásakor azonban csökken az átviteli sebesség. Ezt úgy orvosolhatjuk, hogy útközben úgynevezett repeater segítségével felerősítjük a jelet.
Az imént az 50 Ω -s koaxról szóltunk, ezt elsősorban digitális átvitelre használják. Létezik egy másik fajtája is, a 75 Ω-s, amiről mindjárt részletesen is szót ejtünk. Az 50 Ω-s kábelek elterjedtek ugyan, de egyre inkább optikai kábelekre cserélik őket. Ma már inkább csak a kábeltelevíziózás és a helyi hálózatok terén találkozhatunk velük.

A széles sávú (broadband) koaxiális kábel

Ennek jelentősége az analóg átvitelben van, és leginkább a kábeltelevízióknál használják, például ezen jön az HBO.
Ne tévesszen meg senkit a „széles sávú" kifejezés, ez ugyanis a számítógépes hálózatok esetében mást jelent. Itt minden olyan közeg széles sávú, ahol az átvitel analóg módon zajlik. Mivel ezek a kábelek megfelelnek a kábeltelevíziós szabványnak, 450 MHz sávszélességet biztosítanak, és az analóg átvitel következtében nagy távolságok áthidalására (akár 100 km) is alkalmasak. Ahhoz, hogy egy analóg rendszerben digitális jeleket továbbítsunk, egy olyan csatolófelületre (interface) van szükség, amelyik egy bitfolyamból képes analóg jeleket előállítani, és fordítva.
Mivel ezek széles sávú kábelek, „egyszerre" több minden mehet rajtuk. Azaz egyidejűleg televíziós műsorszórásra és digitális adatok átvitelére is használhatják. A hálózat üzemeltetése azonban csak akkor gazdaságos, ha viszonylag nagy területet (például egy egész kerületet) fed le; ezért erősítőkre van szükség, amelyek rendszeresen erősítik a gyengülő analóg jelet.
Ennek a megoldásnak az a hátránya, hogy az erősítők csak az egyik „irányban" képesek a jelek erősítésére. Mit értünk ez alatt? A kábelezés fatopológiába szervezett; s ha két számítógép között csak egyetlen erősítő van, a fában az alacsonyabb szinten lévő gép nem tud a magasabb szinten lévőnek üzenetet küldeni.
A felmerülő nehézség egyik megoldása a kétkábeles rend-

szer, amikor két ugyanolyan kábel fut egymás mellett. Amikor a számítógép adatot küld, akkor a csomagot az első kábelre helyezi, amely eljuttatja azt a fatopológiába szervezett kábelezés gyökeréhez, a főállomáshoz. Ezt követően a főállomás a második kábelen lefelé kezdi továbbítani az üzenetet a fában. Minden számítógép tehát az első kábelen küld és a második kábelen kap adatokat.
Egy másik megoldás lehet az egykábeles rendszer, ahol csak egyetlen kábel van, viszont az adatok küldése, illetve vétele más-más frekvenciákon történik. A számítógép alacsonyabb frekvenciasávra helyezi a küldeni kívánt adatot, amely eljut a főállomáshoz, az magasabb sávra helyezi, majd újra elküldi az adatot.
Mindenképpen érdemes megjegyeznünk a széles sávú rendszerekről, hogy digitális jelek továbbítására kevésbé alkalmasak. Pusztán azért ejtettünk szót róla, mert ezek ma már rendkívül széles körben kiépített hálózatok: nyugaton a háztartások majdnem kilencven százalékában található kábeltelevíziós hálózat, és itthon is egyre jobban terjed.

Fényvezető szálak

A fényvezető szálak annyiban térnek el a „hagyományos” vezetékektől, hogy az adat fényimpulzus formájában továbbítódik. Az ilyen rendszer három fő részből áll: a fényforrásból, az átviteli közegből és a fényérzékelőből. Vélhetőleg nem szorul bővebb magyarázatra, hogy a fényimpulzus jelenléte a logikai 1-nek, hiánya pedig a logikai 0-nak fog megfelelni.
Az átviteli közeg egy rendkívül vékony üvegszál. Amikor a fényérzékelőre fény jut, az elektromos jelet hoz létre.
A fényforrás ugyanezt csinálja, csak fordítva: amikor elektromos jel érkezik be hozzá, fényjelet bocsát ki magából.
Az ilyen adatátviteli rendszernek megvan az a veszélye, hogy a fény úgymond elszivárog, azaz nem fog megérkezni a célpontba. A fénynek létezik azonban egy jellegzetes fizikai tulajdonsága, mégpedig az, hogyha egy másik anyagba (például a levegőből a vízbe) lép át, akkor megtörik. A törés hajlásszöge a két közeg fizikai jellemzőitől függ, elsősorban azok törésmutatójától. Ha a fény egy bizonyos határszögnél nagyobb szögben érkezik az adott közeg határához, érdekes jelenség következik be: a fény nem lép át a másik közegbe, hanem a közeghatárról „visszapattanva” megmarad abban a közegben, ahonnan indult. Ha tehát egy gyakorlatban is jól használható fényvezető szálat szeretnénk, akkor a fény beesési szögének meg kell egyeznie (vagy nagyobbnak kell lennie) a határszöggel, így a fénysugár az üvegszálban marad. Ilyen módon jelentős veszteség nélkül akár több kilométert is megtehet.
Egy üvegszálban természetesen több fénysugár is haladhat egyszerre, mivel a határszögnél nagyobb szögben érkező fénysugarak is az üvegszálban maradnak. Igaz, minden fénysugár különböző szögben verődik majd vissza. Az ilyen fényvezető szálakat (amelyekben egyszerre több fénysugár is közlekedik) többmódusú szálaknak nevezzük.
A fényvezető szálak másik (és jóval drágább) változata az, amikor az üvegszál átmérője nem nagyobb, mint egy-két fényhullámhossz. Ilyenkor az üvegszál hullámvezető, azaz a fény egyenes vonalban, visszaverődés nélkül száguld. Ennek az előnye akkor mutatkozik meg, amikor jóval nagyobb távolságokba kell gyorsan adatokat eljuttatni.

Fénykábelek

A fényvezető kábel felépítése némileg hasonlít a koaxiális kábelre. Középen egy 50μ átmérőjű üvegmag található. Ez az üveg rendkívül átlátszó, majdnem annyira, mint a levegő. Ezt a magot egy szintén üvegből készült tükröző anyag veszi körül, amit egy műanyagburok véd. Az üvegmagot körülvevő tükröző anyagnak a törésmutatója kisebb, mint a magé, így a fénysugár nem lép ki a magból. Egy kábelen belül általában több fényszálat fognak össze. Felmerül a kérdés, hogy vajon milyen módon lehet ezeket a kábeleket egymáshoz csatlakoztatni? A legkézenfekvőbb megoldás az, ha végeire csatlakozókat szerelünk, és ezek segítségével dugaszoljuk össze a kábeleket. Ezzel csak az a gond, hogy ez a megoldás 10–20%-nyi veszteséget eredményez, ami azért elég jelentős. Előnye viszont, hogy a rendszer könnyen módosítható.
Aki nagyon nagy játékos, azzal is megpróbálkozhat, hogy mechanikai úton ragaszt össze két fénykábelt. Ezt úgy kell csinálni, hogy mind a két szálat egy meghatározott szögben óvatosan el kell vágni, és egymáshoz illesztve őket egy fogó segítségével összeszorítani.
A harmadik módja, hogy két fénykábelből „egy legyen”, ha összeforrasztjuk őket. Vesztességgel ekkor is számolnunk kell, körülbelül tíz százaléknyival.
Az üvegszálon kívül a kábelnek fontos része a fényforrás. Ennek is két fajtája létezik: a LED (Light Emitting Diode) és a félvezető lézer. A LED ugyan sokkalta olcsóbb, hosszabb élettartamú és kevésbé érzékeny a szélsőséges hőmérsékletre, viszont sokkal kisebb adatátviteli sebességgel bír.
A fényvezető másik végén a fényérzékelő vagy fotodióda foglal helyet, amely a fényjelet elektromos jellé alakítja. Ez a művelet sajnos ma még legalább 1 ns-t vesz igénybe, ami azzal jár, hogy az adatsebességet körülbelül 1 Gb/s-ra korlátozza. Meg kell jegyeznünk, hogy ma már üvegszálas technikával ennél jóval nagyobb sebességet is elértünk, a csúcs körülbelül 100 Gb/s körül van. A gond csak az, hogy ilyen gyors adatátvitelt a mai számítógépek még nem képesek feldolgozni.

Fénykábelek kontra rézvezetékek

Ennek a versenynek a győztese egyértelműen a fénykábel, mindjárt ki is derül, hogy miért. Az optikai kábelek egyrészt nagyobb sávszélességgel rendelkeznek, és mivel csak kis csillapításúak, akár 30 km-es távolságra is használhatjuk jelerősítés nélkül.
A másik vonzó tulajdonsága, hogy sem az elektromágneses hullámokra, sem az áramimpulzusokra nem érzékeny.
A fotonok „előnyösebb” fizikai tulajdonságokkal is bírnak, mint az elektronok, például nincs elektromos töltésük.
Az elektronoknak azonban igen, így amikor a kábel belsejében mozognak, kölcsönhatásba lépnek egymással.
A mai világban nagyon fontos, hogy helyváltoztatás közben is kapcsolatba tudjunk lépni másokkal; ilyenkor vezeték nélküli adatátvitel szükséges. A következő részben erről lesz szó.

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.

Hogyan térjünk át Linuxra lépésről lépésre?

Bemutatjuk az online kapcsolattartás lehetőségeit.

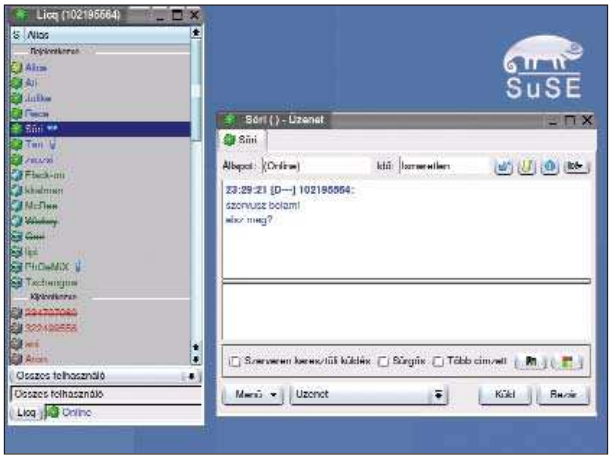
Bizonyára mindannyian hosszú-hosszú órákat töltöttünk el egyes csevegőszolgáltatások előtt ülve, így mindannyian tudjuk, milyen jól el lehet ütni az időt az ilyen kapcsolattartó programok használatával. A jó hír az, hogy erről a remek időrabló szórakozásról Linux alatt sem kell lemondanunk, mi több, lehetőségeink határai egyenesen kitágulnak! Lehetünk a munkahelyünkön vagy az iskolában, de akár otthon is – a háttérben futó alkalmazásokkal kényelmesen tarthatjuk a ismerőseinkkel a kapcsolatot. Jelen cikkben ezeket a Linux alatt elérhető alkalmazásokat szeretném bemutatni a szolgáltatások alapvető ismertetése révén, de a teljesség igénye nélkül, s természetesen ismét a SuSE Linux 9 használatán keresztül.

Üzenetek az interneten

Gondolom, mindenki hallott már az interneten elérhető azonnali üzenetküldő (Instant Messenger) szolgáltatásokról, amelyek révén barátaink elérhetőségét kapcsolatlistánkba (contact list) gyűjtve, szöveges és egyéb üzenetek formájában bármelyikükkel beszélgethetünk. Ha netán valakinek mégsem rémlik, hogy miről is van szó, gondoljon a méltán népszerű ICQ (I Seek You – Téged kereslek) szolgáltatásra, amely az egyik legöregebb és egyben legnépszerűbb üzengető megoldás. Ennek segítségével hasonló módon tarthatjuk a kapcsolatot, mint annak idején egy-egy középiskolai órán teleírt cetliket dobálgatva egymásnak az iskolapadok között. Az ICQ természetesen nem az egyetlen lehetőség az üzengetésre. A másik nagy zászlóshajó az AOL üzenetküldője, az AIM (AOL Instant Messenger), amely szintén jelentős felhasználói bázissal büszkélkedhet, s az iparágba nem olyan régen a Microsoft is bekapcsolódott a saját Microsoft Instant Messenger nevű programjával. Nem nagyon ismerek olyan embert, aki ne használná a fenti programok valamelyikét, és feltételezem, hogy Linuxra áttérve folytatni szeretné a használatukat. Mit tehet a frissen megtért felhasználó? Egyszerűen telepíti az ügyfélprogramokat, bejelentkezik, s élvezi munkája gyümölcsét.

ICQ Linux alatt

Bár az ICQ valamilyen megfontolásból szándékosan nem készít felhasználói számára linuxos ügyfélprogramot, nekünk mégsem kell megijednünk, mert számtalan ICQ-másolat létezik. Ezek közül elsőként a legjobban sikerült és a legjobban hasonlító alkalmazást, a Licq nevű programot szeretném bemutatni. A telepítése többféle módon



1. kép A Licq üzenetküldő

történhet: az egyik lehetőség, hogy a YaST segítségével megkeressük, majd telepítjük a `licq` nevű csomagot. Ebben azonban nincs benne a KDE munkakörnyezethez illeszkedő kde-gui nevű bővítmény (plugin), így néhány látványelem nem úgy fog megjelenni, ahogyan a Windowsban megszokhattuk, de ettől függetlenül tökéletesen használható. Hogy a problémán segítsünk, a Licq honlapjáról (➲ http://www.licq.org) letölthető a legfrissebb SuSE-csomag, amelyben viszont szerepel a kívánt bővítmény.
A letöltött csomagok telepítése az alábbi parancs kiadásával történik: `rpm -i <csomag neve>`. A nagyon kezdőknek azt javaslom, hogy a terjesztés beépített `licq` csomagját a YaST segítségével telepítsék, akik viszont egy kissé hozzáértőbbnek vallják magukat, azok nyugodtan kézzel tegyék fel.
Megjegyzés: a kézi telepítés után a `kde-gui` bővítmény nem alapértelmezett, ezért az első indítás után lépjünk ki, keressük meg a saját könyvtárban lévő ***.licq*** könyvtárban található ***licq.conf*** fájlt, s a `[plugins]` szakaszban írjuk át az első bővítmény nevét `qt-gui`-ról `kde-qui`-ra, majd újból indítsuk el a programot. A beállításokban látni fogjuk, hogy most már például elérhető az a bizonyos ***Dock ikon***, amelynek segítségével a tálcára ültethető a futó program ikonja.

A Licq használata

Az alkalmazás a `licq` parancs kiadásával indítható el. Az első indítás során egy varázsló siet a segítségünkre, amellyel megadhatjuk a bejelentkezéshez szükséges

2. kép A Gaim programhoz tartozó fiókok

3. kép A Gaim bejelentkező képernyője

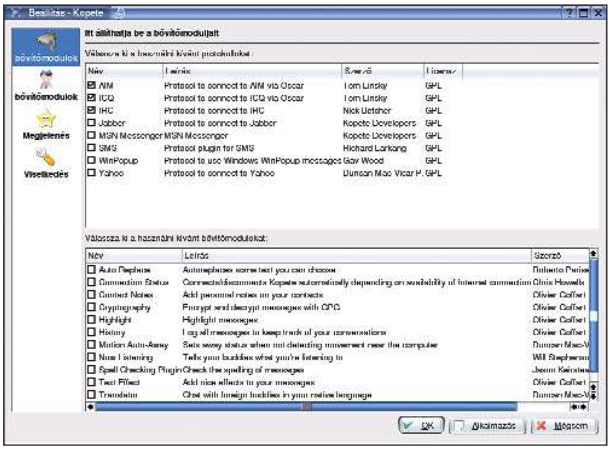
adatokat. Sajnos új felhasználó bejegyzésére nincs lehetőségünk, ezt a weben kell megtennünk – erről a későbbiekben még szó lesz. (Mint tudjuk, a rendszer használatához egy számlára vagy más néven fiókra – account – van szükségünk, amellyel az ICQ rendszerébe bejelentkezhetünk.) A legelső teendők tehát az, hogy beikszeljük a ***Létező felhasználó regisztrálása*** lehetőséget, majd megadjuk az ICQ-számunkat, s a hozzá tartozó jelszót. A befejezés gombra kattintás után nincs más dolgunk, mint bejelentkezni az ablak bal alsó sarkában elérhető menüből. Ekkor az ügyfél az ICQ kiszolgálójáról letölti a kapcsolatlistánkat (contact list). Néha előfordul, hogy az első bejelentkezés során minden ismerőst úgy mutat, mintha nem lennének bejelentkezve. Ez esetben jelentkezzünk ki, majd újra be, s a hiba minden bizonnyal megoldódik. Innentől a program használata a Windows alatt megszokott ügyfélével teljesen azonos. A ***Rendszer*** menü ***Beállítások*** és ***Felületválasztó*** menüpontjának segítségével szabhatjuk testre az alkalmazást.

Fontos tudnivaló, hogy mielőtt kilépnénk, feltétlenül kattintsuk a ***Rendszer*** menü ***Beállítások mentése*** menüpontjára, mert a beállításaink csak így rögzülnek.

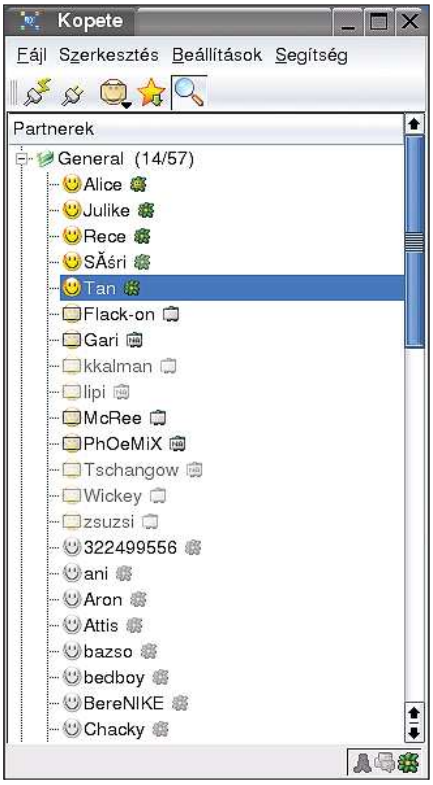
A programmal egyébként az összes ICQ-alapszolgáltatást használhatjuk, de nem tudjuk igénybe venni a különleges szolgáltatásokat (például az ICQPhone-t). Ez egyébként a Windows esetében is külön programcsomagban található meg.

Szöveges ICQ

Mint tudjuk, kedvenc operációs rendszerünk parancssorban rendkívül erős, nem meglepő hát, hogy szinte minden létező grafikus programnak létezik parancssorból vagy



4. kép A Kopete üzenetküldő



5. kép Kopete: ismerősök

távoli terminálról is futtatható változata. Nincs ez másképp az ICQ-ügyfelek esetében sem. Maradjunk még egy kicsit az előző programnál. A Licq alapértelmezetten egy olyan bővítményt tartalmaz, amellyel terminálmódú üzenetváltást valósíthatunk meg az ICQ hálózatán. A bővítmény használatához csak annyi a dolgunk, hogy a saját könyvtárunk ***.licq*** alkönyvtárában található *licq.conf* fájl `[plugins]` szakaszában az alapértelmezett bővítményt `kde-gui`-ról `console`-ra írjuk át, majd parancssorból kiadjuk az `licq` parancsot. Ezt követően a programban /-rel kezdődő parancsokkal vezérelhetjük az üzenetváltásokat. Mindenekelőtt a `/help` parancs kiadását és az ez alapján történő tájékozódást javaslom.

AIM Linux alatt

Az AOL üzenetküldő szolgáltatását a legegyszerűbben a `gaim` nevű csomag telepítésével vehetjük igénybe. A Gaim (Gnome AIM) egyébként az összes üzenetküldő hálózatot kezeli a Yahoo! megoldásától kezdve az ICQ-n keresztül a Microsoft által alkotott üzenetszolgáltatásig.

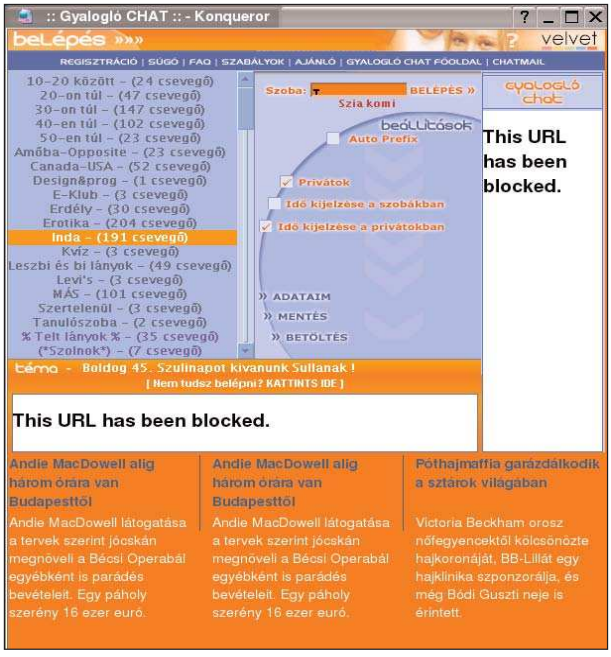
A program telepítése után az első indítás során itt is egy bejelentkező ablak pattan fel, ahol megadhatjuk az AOL-os nevünket és a jelszavunkat. Ha további üzenetküldőket is igénybe szeretnénk venni, akkor válasszuk az ***Eszközök*** menü ***Fiókok*** pontját, ahol a jelenleg beállított fiókokat láthatjuk. A ***Hozzáadás*** gombra kattintva válasszuk ki a megfelelő protokollt (ICQ, AIM, Yahoo, MSN stb.), majd adjuk meg a beállításokat. A ***Mentés*** gombot használva az előbbi listában az újabb kapcsolat is megjelenik. Jelentkezzünk be arra is, így egy időben egyszerre több hálózaton is beszélgethetünk, s egy programmal foghatjuk össze a különböző üzengetőket használó barátainkat. A program használatáról szintén nem szeretnék túl sokat beszélni, hiszen bonyolultságát tekintve megegyezik azokkal, amelyekhez eddig szokhattunk, így biztosan nem jelent nehézséget senki számára.
Természetesen nem ez az egyetlen megoldás. Az AIM például linuxos ügyfélprogramot biztosít felhasználóinak, amely a ➲ http://www.aim.com/get_aim/linux/latest_linux.adp oldalról tölthető le.

Mi történik, ha én új felhasználó vagyok valamelyik hálózaton?

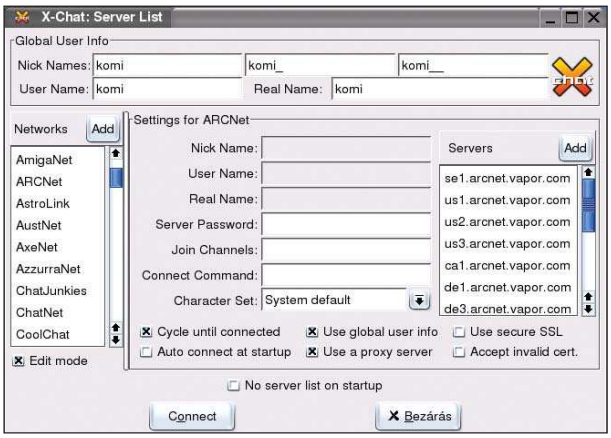
Mint már említettem, egyik Linux alatt elérhető program sem kínál megbízható ügyfélbejegyzési lehetőséget (van olyan, amelyik egyáltalán semmilyet nem kínál, némelyiknél pedig egyszerűen csak nem működik).
Ha azonban mi mégis szeretnénk valamelyik üzenetküldő hálózatba bekapcsolódni, akkor kénytelenek leszünk ezt a szolgáltató weboldalán megtenni. Szerencsére a legtöbben ezt lehetővé teszik, mi több, az AOL által kínált linuxos AIM-ügyfél az új bejegyzés során egyenesen a böngészőt indítja el és a kívánt weblapra kalauzol bennünket. Most azonban nézzük meg, hogy miként nyithatunk számlát (account) a három legelterjedtebb azonnali üzenetküldő-szolgáltatáshoz:
ICQ: ➲ http://web.icq.com/register
AIM: ➲ http://my.screenname.aol.com – Itt kattintsunk az oldal jobb felén található ***Get a Screen Name*** hivatkozásra, majd adjuk meg az adatainkat.
Yahoo!: ➲ http://messenger.yahoo.com – Itt kattintsunk a Yahoo logo mellett található ***Sign Up*** hivatkozásra, majd adjuk meg az adatainkat.

Kopete, avagy a legegyszerűbb megoldás minden gondra

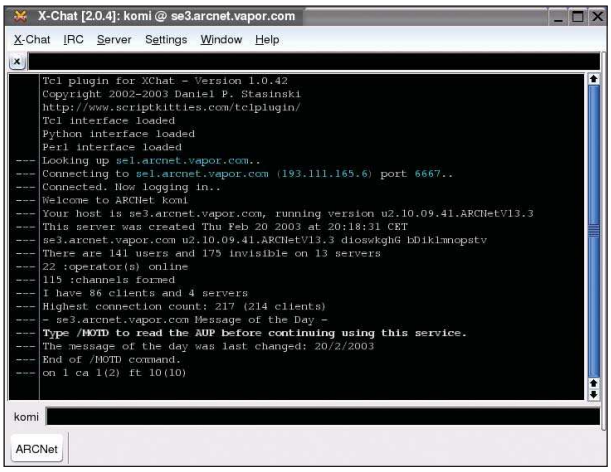
Ha valaki az idők folyamán az összes létező üzengető szolgáltatásra sikeresen rászokott, biztosan elmegy a kedve az élettől, mire mindegyik programot beállítja Linux alatt is. Jó megoldásnak ígérkezik a Gaim használata, amely – mint láthattuk –, az összes jól ismert azonnali üzenetküldő-szolgáltatást kezeli. Létezik azonban egy ennél kifinomultabb program is, amelyről a legtöbb linuxos ismerősöm elismerően nyilatkozik: ez az a bizonyos Kopete nevű alkalmazás. Egyik nagy előnye, hogy a grafikus felhasználói felületek versenyében jelenleg nyerésre álló (de legalábbis az Európában jobban elterjedt) KDE munkakörnyezeten alapul, s a munkakörnyezet részének tekinthető. Olyannyira, hogy SuSE 9 alatt már telepítenünk sem kell, ez az alapértelmezett azonnali üzenetküldő alkalmazás. Másik előnyének mondható, hogy a szolgáltatáshoz történő kapcsolódás tekintetében a Kopete bővítményes jellegű, amely bővítményekhez a program maga felületfüggetlen módon kapcsolódik, ezáltal egy újabb üzenetküldő szolgáltatás hozzáadásához egyetlen bővítmény telepítése elegendő. Kinézetét és használatát tekintve ez sem sokban tér el az eddig megszokottaktól. Az egyetlen dolog, amiről szólnom érdemes, az az egyes szolgáltatásokhoz történő kapcsolódás a fentebb említett modulok, bővítmények segítségével.
Válasszuk a ***Beállítások*** menü ***A Kopete beállításai*** pontját, s nézzünk szét egy kicsit a megjelenő panelen! (Megjegyzés: az alkalmazás első indítása során ez a panel jelenik meg alapértelmezetten.)
Az első, „bővítőmodulok” részben válasszuk ki, hogy milyen protokollokat szeretnénk használni (milyen üzenetküldő szolgáltatásokat szeretnénk igénybe venni), majd a második „bővítőmodulok” részben adhatjuk hozzá az adott szolgáltatást. Ehhez kattintunk az ***Új*** gombra, majd a varázsló lépéseit követve állítsuk be a megfelelő kapcsolódási módot. Külön érdekesség, hogy például ICQ protokoll használata esetében megadhatjuk az ICQ-nál is nyilvántartott egyéb adatokat (lakcím, születési év stb.), amelyet egyébként az ICQ weblapjáról is kezelhetünk.
Ezek után, ha visszatérünk a főablakhoz, annak jobb alsó sarkában minden hálózathoz egy ikont találunk, amelyre a jobb gombbal kattintva febukkan annak a menüje. Mindegyikkel külön-külön csatlakozhatunk, egy időben egyszerre akár több kapcsolat is élhet. Az egyes felhasználókat a különböző hálózatokon úgy tudjuk megkülönböztetni, hogy mindegyikőjük neve mellett az adott üzenetküldő protokoll ikonja szerepel.
Csakúgy, mint a Licq, a Kopete is kezeli a fájlok küldését, fogadását, csevegőszobák létrehozását, s mindkettőbe belekerült a központi kiszolgálón tárolt kapcsolatlisták kezelése is. Ezekkel együtt az üzengető programok legjobbjaivá váltak.



6. kép Az egyik hazai webes csevegőszolgáltatás



7. kép Az X-Chat kiszolgálólistájának szerkesztése



8. kép Az X-Chat nevű IRC-ügyfél

A fent bemutatott programok természetesen csak a legismertebbek. Számtalan egyéb alkalmazás elérhető, s mindegyik más-más irányú és színvonalú szolgáltatást nyújt. Akit érdekel, az legkönnyebben a YaST csomagkezelő moduljának segítségével tájékozódhat az elérhető programokról. A YaST ***Szoftver telepítése és eltávolítása*** modulban a ***Csomagcsoportok*** szűrő alkalmazásával a megjelenő kategóriafában, az ***Alkalmazások/Hálózat*** alkategóriában találhatjuk az általunk keresett programokat.

gnomeICU

A GAIM-mal azonos kaliberű csodabogárról van szó, amely szintén támogatja az AIM és az ICQ hálózatának együttes használatát. A program felülete egyébként a Gnome 2.x-es változatához illeszkedik. A Gnome panelen meg is jelenik az indítás után a dokkoló ikon, amely jelzi állapotunkat, a bejelentkezett barátaink számát, s azt is, hogy van-e olvasatlan üzenetünk. Használatát épp ezért azoknak javaslom, akik a Gnome mellett tették le a voksot, s azóta is azt használják.
Telepítés gyanánt a YaST csomagkezelőjével keressünk rá a *gnomeicu* nevű csomagra, majd adjuk hozzá a rendszerünkhöz. A rövid telepítési folyamat után adjuk ki a `gnomeicu` parancsot, amellyel életre keltjük a programot. Induláskor egy kis ablakot kapunk, amelyen választhatunk, hogy új felhasználók vagyunk-e, avagy korábban már bejegyeztük magunkat. Az üzenetküldő azon kevés linuxos ügyfelek közé tartozik, melynek működik a felhasználó bejegyzését végző modulja. Próbáljuk ki nyugodtan. Alapértelmezetten ICQ-ügyfelet farag belőlünk, s ehhez mindössze a leendő számla (account) jelszavát kell megadnunk, s ezt is csak egyszer. Ezután rögtön elkészül a bejegyzés, ám egyéni adataink módosításához előbb „online” állapotba kell fordítanunk a programot, majd az egyéni beállításoknál adhatjuk meg a becenevünket és minden egyéb más adatunkat is. Mindenképp tegyük meg ezt, mert az alapértelmezett becenév az ***User***, s ezáltal sokan tévesen azonosíthatnak bennünket a barátaink közül. Ha azonban már régi motorosok vagyunk, elég beírni az ICQ-számot és a hozzá tartozó jelszót, s azonnal használatba is vehetjük üzengetőnket. Az első bejelentkezés során a program letölti a kiszolgálóoldali kapcsolatlistát (contact list), s máris használható a szolgáltatás. A program másik érdekessége, hogy igen fürge jószág: minden azonnal, villámgyorsan nyílik. Ehhez némi egyszerűség is társul, de mindez a használat során inkább előnyünkre válik: nem kell kerülgetni a sok színes gombot. Ennek ellenére természetesen ez is támogatja a fájlok küldését, fogadását, egyszóval minden általános szolgáltatás igénybe vehető. Én személy szerint sokáig használtam Gnome 1.x alatt a mostani változat elődjét, de egyetlen panaszos szavam sem lehet vele kapcsolatban.

Csevegőszolgáltatások

Az azonnali üzenetküldés mellett a másik népszerű kapcsolattartási terület a csevegőprogramok használata (chat). A csevegőszolgáltatások eredete korábbra nyúlik vissza, mint az azonnali üzenetküldés, manapság valamiért mégis az utóbbiak váltak népszerűbbé. Itt az emberek nem üzeneteket dobálnak egymásnak, hanem egy csevegőszobákban összegyűlve mindenki mindenkivel beszélget, vagy egyesek párban magánbeszélgetést folytatnak.
Először is mindjárt egy olyan szolgáltatással kezdeném, amelyik nem kapcsolódik szorosan a Linuxhoz, de természetesen sokan kedvenc operációs rendszerükkel csatlakozva is el szeretnék érni a csevegőszobákat. Manapság az egyik legelterjedtebb csevegőforma a weben megjelenő, de a felhasználó gépén futó programok által biztosított csevejfelület. Ez a program az esetek túlnyomó többségében egy Java-applet. Ha pedig Java-appletről van szó, a kezdő linuxos mazsola nem ijed meg, hanem mosolyog, ugyanis tudja, hogy az appletek futtatása gyerekjáték, s a sorozat kettővel ezelőtti cikkében épp arról volt szó, hogy miként okosíthatjuk fel a böngészőnket úgy, hogy futtatni tudja az appleteket. Feltételezve, hogy akkor mindenkinek sikerült a Java futtatókörnyezetet beállítania (SuSE 9 felhasználóknak alapértelmezetten be van állítva), most semmit nem kell tennünk, mindössze ellátogatunk a megfelelő weboldalra. A hazai két legnépszerűbb csevegőoldal a ➲ http://chat.hu, illetve a ➲ http://chat.gyaloglo.hu címen érhető el.

IRC

Az IRC (Internet Relay Chat, nagyjából: interneten továbbított csevegés) az egyik legelső internetes csevegőprotokoll. Bár szabványban rögzített, ez sem teljesen egységes

– különböző hálózatok léteznek sok tízezres felhasználói táborral. A világon számtalan IRC-kiszolgáló található. A webes magáncsevegőkhöz képest azzal a nagy előnnyel rendelkezik, hogy itt az adott kiszolgálóra bejelentkezett felhasználók nemcsak az oda bejelentkezett társalgópartnereiket látják, hanem az abba a hálózatba tartozó összes IRC-kiszolgáló minden ügyfelét is. A mi dolguk tehát annyi, hogy egy ügyfélprogrammal bejelentkezzünk valamelyik tetszőleges, nyilvános IRC-kiszolgálóra. A kiszolgálók adottak, nekünk tehát csak a megfelelő ügyfélprogramot kell biztosítanunk. A cikk további részében két alkalmazásról szeretnék beszélni: az egyik a Mozilla webböngésző része, a másik KDE alapú.

ChatZilla

Mint már említettem, ez az alkalmazás az általunk jól ismert Mozilla webböngésző része. A program futtatása SuSE 9 alatt sajnos csak a Mozilla indításán keresztül lehetséges. (Ha valaki netán rájött, hogyan lehet közvetlenül futtatni, kérem, a tudását ossza meg velem is!) Indítsuk tehát el a Mozilla webböngészőt, majd a jobb alsó sarokban, az állapotsorban található ikonok között keressük meg a ChatZillára mutatót, s kattintsunk rá. Ez alapértelmezetten semmilyen varázslót nem kínál fel, de nekünk az első használat során feltétlenül meg kell nyitnunk a ***Szerkesztés*** menü ***Beállítások*** pontját, s itt meg kell adnunk a becenevünket, a teljes nevünket, s nem árt, ha a leírásban az elektronikus levélcímüket is szerepeltetjük. Hagyjuk jóvá a műveletet az OK gombbal, majd lépjünk ki a programból és a biztonság kedvéért indítsuk újra, mert előfordulhat, hogy nem alkalmazza azonnal az általunk megadott beállításokat. Ha ezzel megvolnánk, akkor legegyszerűbben egy IRC-parancs segítségével csatlakozhatunk a kívánt kiszolgálóhoz. A parancs például az alábbi lehet: ***/server irc.sote.hu***. Ezek után már a szabványos IRC-parancsok segítségével élvezhetjük csevegőalkalmazásunk minden lehetőségét.

X-Chat

Az előző programnál egy sokkal nagyobb tudású ügyféloldali alkalmazás is létezik: az X-Chat. Használatához az xchat nevű csomagot kell telepíteni, s a művelet végeztével a program meglepő módon az `xchat` parancs kiadásával indítható (vagy természetesen a KDE menüből). Az előugró panelen a különböző hálózatok kiszolgálói közül kiválaszthatjuk, hogy mi melyikre szeretnénk kapcsolódni. Ha saját kiszolgálót szeretnénk, akkor kapcsoljunk ***Edit mode***-ba, s adjunk hozzá akár saját hálózatot, akár valamelyik hálózathoz egy új kiszolgálót. Ha ezzel megvolnánk, az adott kiszolgálót kiválasztva kattintsunk a ***Kapcsolódás*** gombra, s ismét abban a helyzetben vagyunk, hogy különféle IRC-parancsokkal folytathatjuk a csevegést.

Ha viszont jobban körülnézünk, láthatjuk, hogy a legfőbb lehetőségeket a menürendszer segítségével is elérhetjük. Beállíthatunk néhány jelzőbitet az IRC menüben vagy éppen különböző műveleteket végezhetünk a ***Server*** menüben. (Az IRC menü egyébként a legalsó menüpont bekapcsolásával tetszőlegesen szerkeszthető.) Ha egy csatorna felhasználólistájában rákattintunk egy becenévre, az előugró gyorsmenü szintén számtalan lehetőséget kínál, ugyanúgy, mint ahogyan például a Mirc nevű windowsos program esetében már megszokhattuk. Lehetőségünk van mindenféle, a hálózattal kapcsolatos adatok lekérésére. Ezeket a listákat a ***Window*** menü különböző pontjain keresztül érhetjük el. A program képességei még messze nem merülnek ki: különböző parancsfájlok futtatása és bővítmények kezelése is módunkban áll, de ezek használatára a kezdeti megpróbáltatások során biztosan nem lesz szükségünk.

BitchX

Ha már azt ígértem, hogy két IRC-es alkalmazásról fogok beszélni, hadd lássa mindenki, hogy milyen ügyesen tudok számolni: jöjjön a harmadik! Az ICQ ügyfélprogramoknál már említettem a Linux parancssoros erejét, s azt is, hogy szinte minden területen létezik terminálmódú megoldás is. A BitchX egy ilyen szöveges környezetben futó IRC-ügyfél, amely valóban egészen „karakteres”-re sikeredett. Egyébiránt teljes értékű ügyféloldali programról van szó, számtalan grafikus társát megelőzi, én azonban mégis csak a legelvetemültebbeknek javasolnám körülményes használatát. Nekünk, kezdőknek elég az is, ha tudomásunk van róla, hogy ilyen is létezik.

Zárszó

Sajnos egyik alkalmazási területen sincs lehetőségem az összes program bemutatására, jelen cikkben is igyekeztem a legérdekesebb, legjobban használható darabok közül válogatni. Ezenkívül számtalan aprócska vagy nagyobbacska alkalmazás létezik, amellyekkel kellő színvonalon elégíthetjük ki további igényeinket. Láthattuk, hogy nem igazán létezik olyan szükséglet, amelyet ne tudnánk a Linux alatt elérhető kapcsolattartó programokkal kielégíteni, így elmondhatjuk, hogy kedvenc operációs rendszerünk az informatika óceánjának ezen a vizén is duzzadó vitorlákkal halad. Sorozatunk következő részében a linuxos képnézegető, illetve képszerkesztő alkalmazásokat vesszük sorra.

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

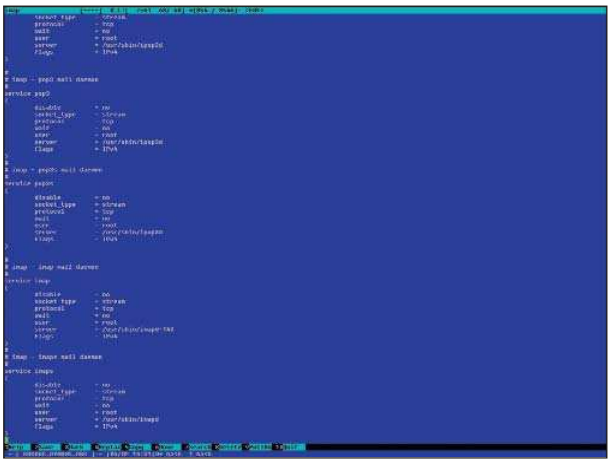
Linuxos kiszolgálót mindenkinek! (5. rész)

A SuSE Linux mint kiszolgáló – kisvállalati és otthoni környezetben.

Cikksorozatom ötödik része elsősorban a különböző levelezési protokollokkal foglalkozik, kiemelt figyelmet szentelve a POP3 és IMAP4 protokolloknak és különböző változataiknak. Cikkem előző részében Postfix SMTP-kiszolgálónkat telepítettük és állítottuk be; most azon leszünk, hogy megfelelő felületet teremtsünk a kiszolgálón a különböző levelezőprogramok csatlakozásához.

Az elmélet

Először nézzük meg a szóban forgó két protokollt! A POP3 (Post Office Protocol version 3) igen népszerű protokoll, a felhasználók jelentős része ezt használja az elektronikus üzeneteknek az internetes kiszolgálókról való letöltésére. A POP3 protokoll segítségével a felhasználó levelezőprogramja letölti a leveleket a kiszolgálóról, majd az esetek nagy többségében törli is onnan őket. A POP3 feladata, hogy a kiszolgálóval kapcsolatot létesítsen, elvégezze a felhasználó hitelesítését, majd a felhasználó alapértelmezett bejövő postafiókjából kiolvassa a leveleket. Ugyanakkor semmilyen módosításra nem képes a kiszolgálón, a levelek rendezése ebben az esetben csak az ügyfél oldalán történhet meg. A működés pontos leírása az RFC 1939 számú leírásban található meg, ami az interneten lelhető fel. (➲ http://www.faqs.org/rfcs/rfc1939.html)
Az IMAP4 (Internet Message Access Protocol version 4) (➲ http://www.faqs.org/rfcs/rfc1730.html) protokoll lehetőséget biztosít arra, hogy a postafiókunkat ne a saját gépünkön, helyben tároljuk, hanem a tartalmát a kiszolgáló őrizze meg számunkra. Ebben az esetben különböző műveletek végrehajtására nyílik lehetőségünk a kiszolgálón, így könyvtárakat hozhatunk létre, leveleket másolhatunk közöttük, újakat hozhatunk létre vagy éppen törölhetjük őket. Ez nagyon hasznos szolgáltatás lehet akkor, ha a felhasználónak arra van szüksége, hogy a világ bármely részéről azonos módon férjen hozzá a postafiókjaihoz. Mindkét protokoll adatfolyam alapú kapcsolatot használ a TCP protokoll felett. A POP3 alapbeállításként a 110-es, míg az IMAP4 a 143-es kaput használja a kapcsolattartásra, így a telepítés után ezeket a kapukat kell engedélyezni a tűzfalon. Mindkét protokoll alapvető hiányossága, hogy a felhasználóhitelesítést és az adatkapcsolatot egyaránt nyílt formában végzi, így a folyamat lehallgatható. Mivel a postafiókhoz tartozó felhasználóknak gyakran egyéb hozzáférésük is van a kiszolgálóhoz, a nevüket és jelszavukat megszerezve a kiszolgáló nagymértékben sebezhetővé válik. Erre kínál megoldást mindkét protokoll esetén az SSL-es (Secure Sockets Layer) változat. A protokolloknak ezek a változatai mind a felhasználói hitelesítés, mind a kapcsolattartás folyamatát titkosított formában végzik, így a támadónak



Az IMAP csomag beállítása a xinetd bejegyzések között

nincs esélye rá, hogy felhasználóneveket és jelszavakat hallgasson le. Ezek a protokollok úgynevezett nyilvános kulcsú titkosítást használnak, aminek nagy előnye, hogy a felépítéséből kifolyólag jelenlegi matematikai tudásunk szerint gyakorlatilag lehetetlen visszafejteni. Ahhoz, hogy ezeket a protokollokat használni tudjuk, úgynevezett tanúsítványok létrehozására van szükségünk, amelyek segítségével a felhasználó és a kiszolgáló közötti adatfolyam titkosításra kerül. Ezeket a tanúsítványokat nekünk kell a telepítés után létrehoznunk, méghozzá eszményi esetben minden kiszolgált tartományhoz külön tanúsítvánnyal. Az utóbbi azért fontos, mert a tanúsítvány a kiszolgáló nevét is tartalmazza, és amennyiben az nem egyezik a megszólított kiszolgáló címével, erről minden esetben figyelmeztetést kapunk. Ez akkor fordulhat elő, ha kiszolgálónknak több neve is van, például ***mail.tartomany.hu*** és ***kiszolgalo.tartomany.hu***. Amennyiben a tanúsítvány a ***mail.tartomany.hu*** címre lett kiállítva, mi viszont a ***kiszolgalo.tartomany.hu*** címen próbálunk csatlakozni hozzá, ügyfélprogramunk erről nagy valószínűséggel figyelmeztetést fog küldeni, hogy felhívja rá a figyelmünket: a kiszolgáló esetleg egy hamis kiszolgáló vagy rossz a tanúsítvá-

nyunk. Mindez azért fontos, mivel a nyilvános kulcsú titkosítás csak addig használható biztonságosan, amíg a tanúsítványok érvényesek és nem sérültek meg. Amint a támadó hozzáfér a tárolt titkos kulcsokhoz és azokkal készít tanúsítványokat, a kapcsolat többé már nem megbízható.
A POP3 SSL-es változata a POP3S, amely a 995-ös kaput használja a kapcsolattartásra, míg az IMAP4 SSL-es változata az IMAPS, amely a 993-as kapun tartja a kapcsolatot az ügyfelekkel.

A gyakorlat

Most, hogy az elméleti alapokkal megismerkedtünk, nézzük a gyakorlatot és telepítsük, valamint állítsuk be a kiszolgálókat! Mind a POP3, mind az IMAP protokoll feladatainak ellátására több kiszolgálócsomag (úgynevezett démon) is létezik, így a POP3-ra az `imap` csomag, a `solidpop3` csomag, vagy akár a `qpopper` csomag. IMAP esetén pedig a legnépszerűbb az `uw-imapd`, amely a SuSE 8.2-es és 9.0-s kiadások alatt az `imap` csomag neve alatt található, valamint itt van a Cyrus IMAPD, amely egy olyan IMAP-kiszolgáló, ami képes a kiszolgálón található felhasználóktól függetlenül is postafiókokat kezelni. Mi most a POP3 és az IMAP4-es protokoll feladatait az `uw-imapd` csomagra fogjuk bízni, így a YaST-tal az `imap` csomag telepítésére lesz szükségünk. Az `imap` csomag futtatásához szükségünk lesz vagy az `inetd`, vagy a `xinetd` Internet Service Daemon (internetszolgáltatás démon) telepítésére, ugyanis ezeket a démonokat csak az adott kapura beérkező kérés esetén indítja el a rendszer. A kérések érkezését pedig az `inetd`, illetve a `xinetd` hivatott figyelni.
A `xinetd` az `inetd` egy továbbfejlesztett, feladatai bővített változata, így ennek a használatát javaslom, amennyiben ez nem jelent lényeges átszervezést a rendszerben. A beállítások leírása során én a `xinetd` használatára támaszkodom.
A csomagok telepítése után a */usr/sbin* könyvtárban megjelenik az `ipop2d`, `ipop3d` és `imapd` futtatható állomány. Ezek lesznek a különböző protokollok kiszolgálódémonjai, amelyeket a `xinetd` internetszolgáltatás-démon indít a megfelelő kapura érkező kérés esetén.
Ezeknek a démonoknak alapvetően nincsenek beállításokra szolgáló állományai, így a csomag telepítése után akár használhatjuk is őket.
Beállítás gyanánt mindössze engedélyezni kell a tűzfalon a megfelelő kapuk elérését, valamint a `xinetd` kiszolgálót szükséges beállítani a kérések megfelelő helyre való továbbítása érdekében.

A nyilvános kulcsú titkosítás

A nyilvános kulcsú titkosítás (Public Key Encoding) alapgondolata, hogy gyakorlati titkosságot tud nyújtani anélkül, hogy a kapcsolattartó partnerek a kapcsolat megkezdése előtt kulcsot cseréltek volna, mint az a hagyományos szimmetrikus titkosítások esetén szükséges. A nyilvános kulcsú titkosító két kulccsal dolgozik: egy nyilvános kp és egy titkos ks kulccsal. A kódoláshoz a nyilvános kulcsot használjuk, míg a dekódolás a titkos kulcs segítségével történik. Ezért is nagyon fontos, hogy a titkos kulcsunkhoz senki ne férhessen hozzá.
Az algoritmus alapját az a matematikai feladvány adja, amely szerint egy szám prímtényezős felbontása algoritmikus értelemben véve nehéz feladat.
Most matekozzunk egy kicsit és nézzük meg, hogy miként is működik az algoritmus! Vegyünk két nagy, akár több száz számjegyű prímszámot, p-t és q-t, továbbá nézzük meg a szorzatukat. Ez nevezzük n-nek. A kódolt szöveget úgy értelmezzük, mint y=xe (mod n), ahol x a kódolandó szöveg, y a kódolt szöveg, e a kódoló függvény, (mod n) az n-nel való maradékosztás jele. Az e kódoló függvényt úgy válasszuk meg, hogy az relatív prím legyen φ(n)-hez, azaz az n szám osztóinak számához. A φ(n) abban az esetben, ha n prímszám, akkor (n-1). Mivel p és q, a két kiinduló számunk prímszám, a φ(n)= φ(p*q)=(p-1)*(q-1). Tehát e-t úgy válasszuk meg, hogy e és (p-1)*(q-1) szám relatív prímek legyenek, tehát (e, φ(n))=1.
Kódolni ezek után már tudunk, most nézzük a dekódolást! Az elv itt is hasonló, egy kongruenciát kell megoldanunk, de most az x=yd (mod n) műveletről van szó, ahol x a kiinduló, tehát immár dekódolt szöveg, y a dekódolandó szöveg, d a dekódoló függvény, valamint (mod n) itt is az n-nel való maradékosztás jele. A d dekódoló függvény és az e kódoló függvény között jelen esetben is összefüggésnek szükséges lennie, mint bármely kódoló–dekódoló felépítés esetén, ez az összefüggés pedig a d*e=1 (mod φ(n)) (tehát a d és e függvény egymás inverze (mod φ(n))). Amennyiben ezt a kongruenciát is megoldottuk, megkapjuk a d dekódoló függvényt, ennek ismeretében pedig már egyszerűen dekódolható az y szöveg.
Az üzenetek kódolásához az n számot és az e kódoló függvényt nyilvánosságra hozzuk, a többit pedig szigorúan titokban tartjuk. Ezek alapján remekül látszik, hogy amíg n-ből nehéz vagy a gyakorlatban lehetetlen előállítani p-t és q-t, tehát nem tudjuk a számot prímtényezőire bontani, addig a titkosítás megfelelő méretű p és q választása esetén gyakorlatilag feltörhetetlen. Hiszen amíg n-t nem tudjuk felbontani, nem bírjuk meghatározni a φ(n)-t, így nem vagyunk képesek meghatározni a d dekódoló függvényt sem.
Ezt a titkosítást mára rengeteg helyen használják, így az összes számítógépes protokoll SSL-es kiterjesztése is, ilyen például az IMAP SSL, POP3 SSL, HTTPS vagy akár az SSH is.
A kódolófolyamat sajátossága, hogy az algoritmus remekül használható digitális aláírás készítésére, hiszen ha megnézzük, hogy a feladó a saját titkos kulcsával írja alá a levelét, akkor az aláíró nyilvános kulcsa birtokában bárki ellenőrizheti a feladó kilétét.
Az eddigieket összefoglalva elmondhatjuk, hogy a kapcsolattartáshoz az egyik fél egy nyilvános kulcsot és egy titkos kulcsot készít, majd a nyilvános kulcsot elérhetővé teszi, például egy weblapon. Amennyiben egy feladó titkosított üzenetet szeretne küldeni nekünk, letölti a nyilvános kulcsunkat, és ezzel kódolja az üzenetet. Ezt a kódolt üzenetet csak a nyilvános kulcshoz tartozó titkos kulccsal fogjuk tudni dekódolni, ezért csak a címzett teheti meg.

A xinetd beállítása

A `xinetd` démon beállítását a */etc/xinetd.conf* állományon keresztül végezhetjük el. Az állomány tartalmazza a démon alapvető beállításait, mint például a naplózás helyét, a kérést indítható ügyfelek címét, címtartományát, valamint a kéréseket fogadó hálózati csatoló címét. A SuSE Linux csomagjából telepített `xinetd` csomag a különböző démonokra vonatkozó beállításokat a */etc/xinetd.d* könyvtárban tárolja. A */etc/xinetd.conf* állományban erre való utalásként az alábbi sor található meg:

```
includedir /etc/xinetd.d
```

Ennek a sornak a jelentése annyi, hogy a –*/etc/xinetd.d* könyvtár alatt található állományok tartalmát a rendszer a beállításállomány futtatásakor fűzze a beállításfájlokhoz. Ennek a megoldásnak annyi értelme van, hogy a beállítások áttekinthetők, mert egyesével, külön állományban vannak elhelyezve. Ennek ellenére mi akár közvetlenül a */etc/xinetd.conf* állományba is készíthetünk bejegyzéseket, bár a rendszer által alkalmazott megoldás sokkal áttekinthetőbb.
A különböző beállítások tehát a */etc/xinetd.d* könyvtárban lévő állományokban találhatóak; bennünket ezek közül most az *imap* állomány érdekel.
Az *imap* állományban találhatóak az `imap` csomag által telepített démonok beállításai, így itt tudjuk a POP2, POP3, POP3 SSL, IMAP4 és IMAP4 SSL démonokat beállítani.
Az összes bejegyzés a következő sorral kezdődik:

```
service <szolgáltatás neve>
```

Itt a szolgáltatás neve a */etc/services* állományban megtalálható szolgáltatásokhoz tartozó kapu számát határozza meg, például a POP3 a 110-zel is helyettesíthető lenne.
A kapcsos zárójelen belül található kapcsolók közül az első a `disable`. Ennek *yes* értéke esetén a démon a kapura érkező hívás esetén sem indul el, ugyanis le van tiltva. Amennyiben egy-egy szolgáltatást szeretnénk használni, ezt az értéket állítsuk *no*-ra. A következő kapcsoló a `socket_type`, ez a kapcsolat típusát adja meg; ne nyúljunk hozzá, hagyjuk `stream` (folyam, adatfolyam) értéken. A `protocol` kapcsoló értelemszerűen a protokoll típusát adja meg, ez `tcp`, ne állítsuk át!
A `wait` és `user` kapcsolókat se piszkáljuk, kivéve, ha például a démont nem rendszergazdaként akarjuk futtatni. Ehhez viszont a különböző futtatási és tárolási állományoknál is gondoskodni kell a megfelelő biztonsági beállításokról, hozzáférési jogosultságokról. A `server` kapcsoló adja a kiszolgáló futtatási állományát, erre még szükségünk lesz, ezért jegyezzük meg. A `flags` kapcsoló az IP-hálózat változatát adja meg, ezt se piszkáljuk, hacsak nem IPv6-os hálózattal rendelkezünk.
Alapesetben számunkra az IMAP SSL és a POP3 SSL csatorna lesz érdekes, mivel a legtöbb levelezőprogram ezeket támogatja, valamint titkosított felhasználóazonosítást és adatátvitelt tesznek lehetővé. Így tehát el kell készítenünk egy `service imaps` és `service pop3s` bejegyzést, ahol a `disable` kapcsolót *no* értékre kell állítani.
Amennyiben az SSL nélküli protokollváltozatokat is használni szeretnénk, az előbb említett `disable` kapcsolót az IMAP és POP3 szolgáltatások beállításánál is állítsuk *no* értékre.
A beállítások végeztével a beállításokat a `/etc/init.d/xinetd restart` paranccsal tölthetjük be. Amennyiben a tűzfalunkon a megfelelő kapuk nyitva állnak, a szolgáltatásokat akár ki is próbálhatjuk.

Nem ilyen egyszerű az élet…

Mielőtt a jól végzett munka édes gyümölcseit élvezvén hátradőlnénk, a `telnet` paranccsal próbáljuk meg megnyitni a beállított szolgáltatásokat. A kiszolgálón egy konzolt indítva a `telnet localhost 110` (ez a POP3), a `telnet localhost 143` (ez az IMAP), a `telnet localhost 993` (ez az IMAP-SSL), illetve a `telnet localhost 995` (ez pedig a POP3-SSL) paranccsal próbálhatunk meg csatlakozni a kiszolgálóhoz. A POP3- és IMAP-kiszolgálókhoz való csatlakozás valószínűleg könnyűszerrel menni fog, és POP3 esetén a következő üzenetet fogjuk kapni:

```
kackac:/etc/xinetd.d # telnet localhost 110
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+OK POP3 localhost v2001.80 server ready
```

Ekkor bejelentkeztünk a kiszolgálóra – úgy tűnik, minden működik. A CTRL+] billentyűkombinációval ki is léphetünk. IMAP esetében:

```
kackac:/etc/xinetd.d # telnet localhost 143
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
* OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS
STARTTLS LOGINDISABLED] localhost IMAP4rev1
2002.332 at Sun, 15 Feb 2004 19:00:21 +0100 (CET)
```

Úgy tűnik, ide is sikerült bejelentkezünk, de ne örüljünk annyira. Éles szemű olvasók észrevehették a `LOGINDISABLED` szócskát a kiszolgáló kimenetén. Nos, biztonsági megfontolásokból a SuSE 8.2-es és 9.0-s kiadásokba épített IMAP-kiszolgálókra csak az SSL-es protokollt használva lehet bejelentkezni. Ugyanakkor felmerül a kérdés, hogy mit tegyünk, ha ez nekünk nem elegendő, mert például a *SquirrelMail* webes levelezőrendszert szeretnénk használni, ami nem ismeri az IMAP protokoll SSL-es kiterjesztését. Ez ugye elvileg nem is jelent biztonsági kockázatot, mert ugyan a csatorna nem használ titkosítást, ellenben nem is jut ki a kiszolgálóról, hiszen mind a webes levelezőrendszer, mind pedig az IMAP-kiszolgáló ugyanazon a gépen található.
Mi a megoldás? Mint fent írtam, ez a 8.2-es és 9.0-s rendszerekben bevezetett korlátozás. Keressünk tehát egy 8.1-es csomaggyűjteményt, például az ➲ ftp.suselinux.hu kiszolgálón, töltsük le az *ftp.suselinux.hu/8.1/suse/i586* könyvtárból az *imap-2001a-155.i586.rpm* csomagot, majd egy Midnight Commander (mc) segítségével lépjünk be az *rpm* állományba. Ezen belül a */usr/sbin* könyvtárban egy `imapd` futtatható állományt találunk, ő lesz a mi barátunk. Másoljuk be ezt az állományt, mondjuk *imapd-143* néven a

/usr/sbin könyvtárba, módosítsuk a */etc/xinetd.d/imap* állományban az IMAP szolgáltatáshoz tartozó beállítások között a `server` kapcsolót */usr/sbin/imapd-143* névre. Ezek után mentsük az állományt és a már megismert módon indítsuk újra a xinetd kiszolgálót.
Láss csodát, ezek után eltűnik a `LOGINDISABLED` felirat a kiszolgáló címsorából. Nyertünk.
Figyelem, azért nem a teljes `rpm` csomag telepítését javasoltam, mert az felülírja a mostani csomagunkat és a 8.1-es változathoz tartozó IMAP-démon pedig az SSL-es IMAP protokollt nem ismerte.
Figyeljünk oda rá, hogy a titkosítás nélküli protokollokat lehetőleg csak a legutolsó esetben használjuk, illetve akkor, ha meg vagyunk győződve róla, hogy biztonságos környezetben tesszük (például a `localhost`-on vagy saját vállalati, illetve otthoni hálózatunkon belül).
Na, akkor most vegyük sorra a POP3 SSL és IMAP SSL protokollokat!
Mind a 993-as, mind a 995-ös kapura való bejelentkezéskor az alábbi megy végbe:

```
kackac:/etc/xinetd.d # telnet localhost 995
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Connection closed by foreign host.
kackac:/etc/xinetd.d #
```

Azonnal visszakapjuk a parancssort. „Hmmm…”, gondolhatjuk, itt valami nem felel meg. Gyorsan utánanézünk a */var/log/mail* naplóállományban, hogy mi lehet a baj.
Itt a következőt találjuk:

```
Feb 15 19:18:32 kackac ipop3d[4535]: Unable to
➥load certificate from /etc/ssl/certs/ipop3d.pem,
➥host=localhost [127.0.0.1]
```

Na igen, megvan a hiba. Korábban beszéltünk már a tanúsítványokról, amikre az SSL protokollok működéséhez szükség van. Nos, pont ezt a tanúsítványt hiányolja a kiszolgáló. De semmi vész, mindjárt létrehozzuk a megfelelő tanúsítványokat.

Hitelesítésszolgáltatás röviden

Linux-rendszerek alatt az `openssl` csomag segítségével hozhatunk létre Certification Authority-t, azaz hitelesítésszolgáltatót. Az ilyen hitelesítésszolgáltató rendszer feladata a fő tanúsítvány létrahozása, illetve új tanúsítványok kibocsátása, kezelése és adott esetben visszavonása.
A hitelesítésszolgáltató rendszerek egy állományrendszerhez hasonlóan faszerkezetben épülnek fel. Legfelül a fő tanúsítvány áll, ami különleges a rendszerben, mivel ez az egyetlen olyan tanúsítvány, amely saját magával van aláírva. Az összes alsóbbrendű tanúsítvány úgy készül, hogy a hierarchiában egy feljebb álló tanúsítvánnyal kerül aláírásra, így igazolva annak hitelességét. A szerkezet felépítéséből látszik, hogy akárcsak a nyilvános kulcsú titkosításnál a két prímszámunkra, úgy itt a fő tanúsítványra kell nagyon vigyázni. Ha a fő tanúsítvány illetéktelenek számára hozzáférhetővé válik, az összes általunk kibocsátott tanúsítvány hitelessége elveszett.
Nézzük, miként tehetjük elérhetővé rendszerünk IMAP és POP3 SSL szolgáltatásait. Pillanatnyilag nem foglalkozunk a hitelesítésszolgáltató rendszerrel, létrehozunk egy tanúsítványt, amit azok a felhasználók fognak használni, akik megbíznak a mi rendszerünkben. Ez a tanúsítvány a későbbiekben majd lecserélhető lesz a hitelesítésszolgáltató rendszerből kibocsátott tanúsítvánnyal.
Először is keressük meg, hogy a rendszerünk melyik könyvtárban tárolja az elkészített tanúsítványokat – SuSE alatt ez általában a */etc/ssl/certs* könyvtár. Váltsunk tehát ebbe a könyvtárba, majd adjuk ki a következő parancsot:

```
openssl req -new -x509 -nodes -out imapd.pem
➥-keyout imapd.pem -days 365
```

Fontos, hogy a készítendő tanúsítvány adatainak kitöltésekor a *Common Name* (CN) mindenképpen az a név vagy IP-cím legyen, ami alapján hivatkozni szeretnénk a kiszolgálóra, ellenkező esetben a tanúsítvány hibát jelezhet.
A `days` kapcsoló állításával tudjuk állítani, hogy a tanúsítvány milyen hosszú időtartamra legyen érvényes. A 365 természetesen pontosan egy évet jelent.
Amennyiben több tartományt is ki szeretnénk szolgálni, ezek megkülönböztetésére készítsünk különböző tanúsítványokat. Ezt úgy tehetjük meg, ha a fenti parancsot a következőképpen módosítjuk:

```
openssl req -new -x509 -nodes -out
➥imapd-sajat.tartomany.hu.pem -keyout
➥imapd-sajat.tartomany.hu.pem -days 365
```

vagy

```
openssl req -new -x509 -nodes
➥-out imapd-XXX.XXX.XXX.XXX.pem
➥-keyout imapd-XXX.XXX.XXX.XXX.pem -days 365
```

Ebben az XXX.XXX.XXX.XXX a kiszolgáló IP-címe.
Ha elkészültünk a tanúsítvánnyal, a következő paranccsal olyan formába alakíthatjuk át, amelyet a legelterjedtebb ügyfelek fel tudnak dolgozni:

```
openssl x509 -in tanusitvany-neve.pem
➥-out tanusitvany-neve.crt
```

Mostani cikkemben rendszerünket felkészítettük a különböző levelezőügyfelek fogadására és elkészítettük a szükséges tanúsítványokat. Mindenki próbálgassa a rendszert, de tartsuk szem előtt az alapvető biztonsági szempontokat. Lehetőleg ne nyissunk felesleges réseket a tűzfalon és ne forgalmazzunk kódolatlan formában jelszavakat!
Sok sikert mindenkinek!

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

Pingvin-fortélyok

Bevált trükkök Linux alatt mindennapjaink megkönnyítésére.

A napokban kért meg egy barátom, hogy telepítsek fel egy Debian Woody-t a számítógépére. Mivel csak most ismerkedik a Linuxszal, fontos volt, hogy az alaprendszer telepítésén túl azokba a mindennapi munkát segítő tippekbe is beavassam, amelyek hosszú kísérletezgetések eredményeként váltak tudásom részévé. Elmondta, hogy nem szeretné végigjárni ugyanezt az utat a varázslóvá válás érdekében. Álljon itt néhány titok azoknak, akik csak most fedezik fel azt az élményt, amit a Linux jelent. Csak a kezemet figyeljétek!

Top a 6-oson

Bizonyára te ismered a `top` nevű, remek segédprogramot, ami a */proc* állományrendszer adatait felhasználva listázza ki a legerőforrás-igényesebb folyamatokat. Ez a szép összesítő jól mutatna az egyik konzolon. A legkézenfekvőbb megoldás természetesen az, hogy belépünk a 6-oson és kiadjuk a `top` parancsot. Ennek számos hátránya van amellett, hogy nem szép megoldás: egyrészt újraindítás után újra el kellene kézzel indítani a `top`-ot. Másrészt, ha felállunk a számítógép mellől, egy nem túl szakavatott felhasználó is odaléphet, megnyomhatja a q-t (kilépés), és kap egy parancssort.
Felmerül a kérdés: nem lehetne-e a `top`-ot is ugyanúgy indítani, mint a */bin/login*-t? Nem tettem volna fel a kérdést, ha a válasz nem lenne egyértelmű: de igen! Kukkantsunk bele a */etc/inittab*-ba:

```
6:23:respawn:/sbin/getty 38400 tty6
```

Ez az `init` beállításállománya. Az `init` az első program, amelyik egy Linux-rendszeren elindul, ezért mindig az 1-es folyamatazonosítót kapja. Az ő feladata befűzni az állományrendszereket, elindítani a különböző démonokat és nem utolsó sorban előkészíteni a konzolokat a belépésre. A fent látható sor a 6-os konzolon teszi lehetővé a bejelentkezést. A : (kettősponttal) elválasztott mezők közül az első egy tetszőleges azonosító. A második mutatja, hogy mely futási szinteken kell elindítani. A harmadikban szereplő `respawn` utasítja az `init`-et arra, hogyha a program véget ér, indítsa újra (élessze fel). Az utolsó mező a számunkra is fontos rész, itt található ugyanis a `getty`. Ez az utóbbi egy terminált nyit meg és egy tetszőleges programot futtat rajta, alapértelmezésben a */bin/login*-t. Mi azonban arra kérjük, hogy inkább a `top` induljon el:

```
6:23:respawn:/sbin/getty -i -l /usr/bin/top
➥-n 38400 tty6
```

A `-i` kapcsolóval érhetjük el, hogy a `getty` ne írassa ki a */etc/issue* állomány tartalmát a `top` indítása előtt a képernyőre. A `-l` használata egyértelmű, míg a `-n` arra utasítja a programot, hogy ne kérjen felhasználói nevet. A változtatások akkor lépnek érvénybe, amikor az `init` újraolvassa az `inittab`-ot, és a 6-os konzol `getty`-je véget ér.

```
# kill -HUP 1
# ps aux | grep tty6
```

A második parancs kimenetéből böngészd ki a 6-os konzolhoz tartozó `getty` folyamatazonosítóját, és lődd ki a `kill` segítségével.
Ezzel azonban még nem vagyunk készen. A `top` ugyanis most többet tud, mint amennyit kellene. Ebből az alkalmazásból közvetlenül jelzéseket lehet küldeni a folyamatoknak. Ha így hagynánk ott a gépünket, a 6-os konzolon ez a felirat üvöltene: „Gyere és lődd ki a rendszergazda folyamatait a `top` segítségével!" Ennek elkerülésére a `top`-nak létezik egy biztonságos módja. Ha parancssorból egy `s` kapcsolóval indítod, nem fogja engedni a `kill` használatát. Ahhoz, hogy a `top` mindig biztonságos módban induljon el, globális beállításállományát, a */etc/toprc*-t kell szerkeszteni. Ha ez az állomány egy `s`-en és egy soremelésen kívül mást nem tartalmaz, már elégedettek lehetünk. Fut egy biztonságos `top` a 6-os konzolon, mely kilépés és újraindítás után is megmarad – ez volt a célunk.

Rendszergazda vagyok, de nem tudom a jelszót!

Biztosan te is hallottad már a nagy bölcseket, amikor óva intettek attól, hogy folyamatosan rendszergazdaként használd a gépet. Valóban nem tanácsos így tenni, mivel ekkor az összes általad indított folyamat is korlátlan jogosultságokkal bír. Csak a legszükségesebb esetben érdemes rendszergazdai parancssort indítani. A `su` pont erre való. Én legtöbbször a – (mínuszjellel) indítom, aminek hatására a kért felhasználó környezetében találjuk magunkat. A jelszavakkal azonban mindig gondban vagyok. Ha túl egyszerű, akkor más is rájöhet; ha túl bonyolult, sokáig tart begépelni. Nem lehetne, hogy az én mindennapi felhasználói fiókom varázsütésre rendszergazdaivá lépjen elő? Semmi sem lehetetlen, ha a

pingvin ereje veled van. A megoldást a PAM (Pluggable Authentication Modules) jelenti.
A PAM megjelenése választotta szét az azonosítást megkövetelő alkalmazásokat magától az azonosítás folyamatától. Egy olyan programnak, amely távoli bejelentkezést tesz lehetővé, nem kell foglalkoznia azzal, hogy a felhasználó hitelesítése jelszóval vagy ujjlenyomattal történik-e. Csupán megkérdezi a PAM-ot, hogy átengedje-e vagy sem. A PAM-ot pedig megfelelő jogosultságok birtokában mi, mezei rendszergazdák állíthatjuk be.
Régen a */etc/pam.conf* tartalmazta az összes szolgáltatás beállítását. Ma már ezek külön állományokban találhatók a */etc/pam.d* könyvtár alatt; minden fájl egy-egy szolgáltatáshoz tartozik. Számunkra most a */etc/pam.d/su* lesz érdekes:

```
# This allows root to su without passwords
➥(normal operation)
auth       sufficient pam_rootok.so
```

Ha a rendszergazdának lehet, nekünk miért nem? Lehet, csupán találnunk kell egy olyan modult, ami felhasználók egy csoportját tudja engedélyezni.

```
auth       sufficient pam_listfile.so onerr=fail
➥sense=allow file=/etc/su_ok item=ruser
```

A *pam_listfile.so* modul egy egyszerű szöveges állományt vesz alapul. Ebben minden sor egy-egy felhasználói név. Az `onerr` kapcsoló azt mondja meg, hogy a fájl megnyitásakor előforduló hiba esetén a `sense` által meghatározott hatás legyen-e érvényben. A mi beállításunk röviden annyit jelent, hogy ha a fájl nem található, akkor nem ért véget a hitelesítés, s ugrik a következő sorra (amely majd a *pam_unix.so* modullal a jelszót fogja kérni). A `sense` kapcsoló a hitelesítés sikeressége esetén kiváltandó hatást mondja meg. Ez azt teszi, hogyha a fájlban megtalálja a felhasználót, akkor engedélyezi a jelszó nélküli `su`-t.
A `file` kapcsolóval az állomány nevét adjuk meg, az `item` pedig a fájlban található sorok jelentését adja meg. Nemcsak a felhasználók neveit vehetnénk ide fel, hanem például csoportokat is, de a kettőt nem lehet keverni. A `ruser` a remote user (távoli felhasználó) rövidítése; azért távoli, mert nem azt a felhasználót keressük a fájlban, akinek a jogosultságait a `su`-val meg akarjuk szeretni, hanem azt, aki a `su` parancsot kiadta.
Végül mindössze egy */etc/su_ok* állományt kell létrehozni, és soronként felvenni azoknak a felhasználóknak a neveit, akiknek engedélyezni szeretnénk a jelszó nélküli belépést. Ezt az állományt – mondanom sem kell – nagyon jól kell védeni, ezért a létrehozás után az az első, hogy kiadjuk a `chmod 600 /etc/su_ok` parancsot.

Vipw, de nem vi

Az `EDITOR` környezeti változó tartalmazza az alapértelmezett szövegszerkesztő elérési útját. Sok alkalmazás használja ezt a változót, többek között a `vipw` is, amellyel a */etc/passwd* tartalma módosítható. Közvetlenül azért nem illik ezt az állományt szerkeszteni, mert a minden körültekintést mellőző felülírással a szerkesztés közben bejelentkezni próbáló felhasználók átmenetileg összeférhetetlenséget tapasztalhatnak. A `vipw` erre megoldást kínál, és az `EDITOR` által meghatározott szerkesztővel kényelmesen módosítható az állomány tartalma. Debian alatt az `EDITOR` alapértelmezésben nincs beállítva. A Bash-megfelelő héjak globális beállítási állománya a */etc/profile*. Ehhez a következő sort hozzáadva a legközelebbi belépéskor már minden felhasználónak a `joe` lesz az alapértelmezett szövegszerkesztője:

```
export EDITOR="/usr/bin/joe"
```

Ezek után a `vipw`-t kiadva már `joe`-ban dolgozhatunk.

Midnight kontra Norton

A jó öreg Norton Commanderen felnőtt felhasználók sokat panaszkodnak amiatt, hogy ha `mc`-ben egy bonyolult elérési úthoz lépnek, majd kilépnek a fájlkezelőből, nem ott találják magukat, ahová elbarangoltak, hanem ott, ahonnan indultak. Ez nagyon egyszerűen orvosolható: Debian alatt az `mc` csomag egy példaparancsfájlt tartalmaz, ezt felhasználva könnyedén megoldhatjuk a gondot. A */usr/lib/mc/bin/mc.sh*-ban a következő áll:

```
mc ()
{
        mkdir -p $HOME/.mc/tmp 2> /dev/null
        chmod 700 $HOME/.mc/tmp
        MC=$HOME/.mc/tmp/mc-$$
        /usr/bin/mc -P "$@" > "$MC"
        cd "`cat $MC`"
        rm -f "$MC"
        unset MC;
}

export -f mc
```

Ez egy `mc` nevű függvényt hoz létre, bizonyos értelemben felülírva az eredeti mc nevű futtatható állományt. Ez azt jelenti, hogyha a fenti parancsfájlt végrehajtjuk, akkor az `mc` parancsot kiadva nem érjük el közvetlenül az áhított programot, hanem ez a függvény fut le. Ami mindössze annyit tesz, hogy az `mc`-t a `-P` kapcsolóval indítja, s ilyen módon az futásának végeztekor visszaadja a legutolsó munkakönyvtárat, és így a `cd` paranccsal lépünk bele. Természetesen ezt a függvényt is minden egyes héj indításakor létre kell hozni, így a */etc/profile* állományba fel kell vennünk a következő sort:

```
. /usr/lib/mc/bin/mc.sh
```

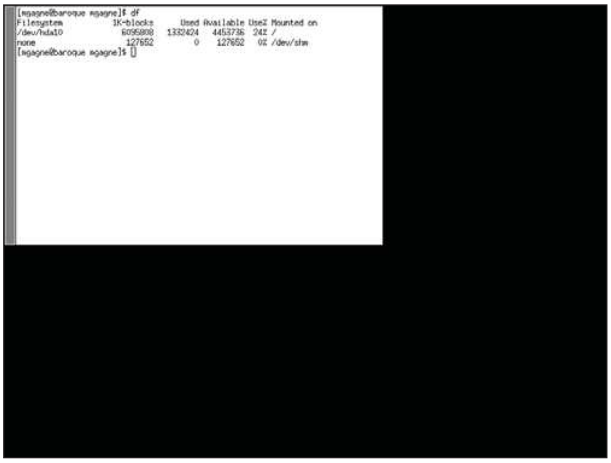
A . (pont) ebben az esetben külső parancsfájl meghívását eredményezi.

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

Munkaasztalból soha nem elég!

Fedezzük fel az új asztali környezeteket anélkül, hogy szokásos felületünket ehhez el kellene hagynunk. Szakácsunk azt mutatja be, hogyan ágyazhatunk egymásba X-munkafolyamatokat.

François, úgy látom, a Window Maker használata mellett döntöttél – nagyszerű! Az egy remek ablakkezelő, élvezni fogod vele a munkát. Két héttel ezelőtt a Gnome-ot futtattad, a múlt héten a KDE volt terítéken, most pedig a Window Maker. Örülök, hogy megfogadtad a tanácsomat és más ablakkezelőket is kipróbálsz. Végtére is a Linux használatának az egyik legnagyszerűbb vonása, hogy mindig van választási lehetőségünk.
Quoi? Hogy én melyiket használom? Ezek mindegyikét és még vagy féltucatot, mon ami! Nem, François, nem ugratlak. Az elsődleges munkafelületem a KDE, de itt van a Gnome, amott az XFCE, és azon a virtuális asztalon a Window Makert futtatom. Sőt Window Maker-munkafolyamatomban még az IceWM is fut. Nem, mon ami, egyáltalán nem bonyolult a dolog, és meg is fogom mutatni neked, mihelyt megérkeznek a vendégeink. De már itt is vannak. François! Vite! A pincébe, gyorsan! Mivel a mai menü inkább desszert, mint főétkezés, menj le a 2001-es Niagara Peninsula rizling jégborért és hozd fel tout de suite!
Isten hozott titeket Chez Marcelnél, foglaljatok helyet! Éppen arra próbálom rávenni hűséges pincérünket, hogy próbálja ki a különféle munkaasztalokat (desktop). Mindig érdekes próbára tenni valami újat, ezért változtatjuk a menüt is állandóan, non? Ugyanez áll az asztali környezetünkre is. Lehet, hogy a KDE vagy a Gnome a kedvencünk, de miért ne próbálhatnánk ki a Window Makert, az IceWM-et vagy az XFCE-t egy kis változatosságért? Ha ellátogatunk ***Matt Chapman*** Window Managers for X (ablakkezelők az X rendszerhez) honlapjára, valószínűleg megjön az étvágyunk egy kis ablakkezelő-változtatásra. Sőt miért ne próbálhatnánk ki ezeket kedvenc munkaasztalunk futtatása közben? Non, mes amis, nem arról van szó, hogy túl sokáig kóstolgattam a bort. Tényleg lehetőségünk van arra, hogy egyszerre több asztali környezetet futtassunk, és ha belejövünk, a dolog nagyon élvezetes tud lenni.
Ahogy a nyílt forrás világában már hozzászokhattunk, természetesen ennek is több módja létezik. Az egyik, hogy a pillanatnyi X-folyamatunkból az egyik virtuális terminálunkra lépünk át. Ha már futtatunk egy X-folyamatot, nyomjuk meg a CTRL-ALT-F1 billentyűkombinációt – ennek hatására újra a szöveges képernyőn kell találnunk magunkat. Egyébként az F1 ugyanígy lehetne akár F2, F3, F4, F5 vagy F6. Amennyiben az X-et a parancssorból indítottuk (és nem valamilyen beléptetőprogramból, mint a `gdm` vagy a `kdm`), látnunk kell a képernyőn a munkafolyamattal kapcsolatos párbeszédet, ahogy megnyomjuk a CTRL-ALT-F1 billentyűkombinációt és az X eseménynaplójának a kimenetét. Az F2–F6 funkcióbillentyűk bármelyikének egy másik szöveges beléptető-képernyőt kell biztosítania; ehhez a második virtuális terminál megjelenítéséhez csak a CTRL-ALT-F2-t szükséges lenyomnunk és így tovább sorban a többit.

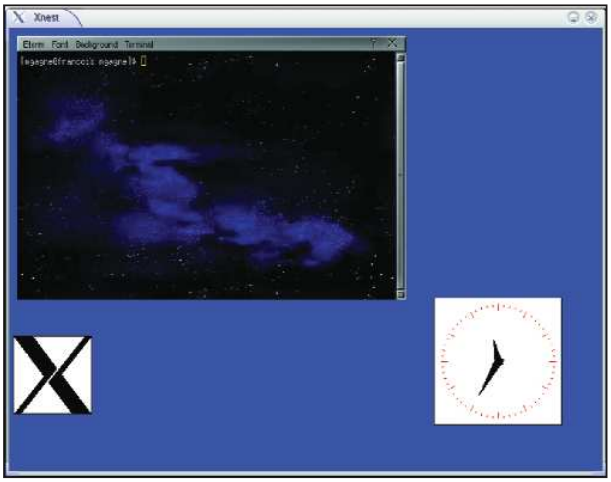


1. kép Egy meglehetősen nehézkes és csupasz X-képernyő egyetlen X-terminállal

Grafikus munkafolyamatunk még mindig aktív. Alapértelmezésben a rendszer által `display :0` névvel jelzett kijelzőn fut, amit – grafikus folyamatunk egyik héjparancsjelébe (shell prompt) az `echo $DISPLAY` parancsot begépelve – magunk is ellenőrizhetünk. Ennek hatására számítógépünk nevének és a kijelzőutótagnak kell megjelennie. Nyomjuk meg a CTRL-ALT-F7 billentyűkombinációt az eredeti (KDE, Gnome vagy valamelyik más) folyamatunkba való visszatéréshez. Rajta, próbáljuk meg, majd térjünk vissza egy szöveges képernyőre (CTRL-ALT-F?)! X-folyamatunk tehát a hetes virtuális terminálon van. A szöveges képernyőn lépjünk be a saját nevünkön és adjuk ki a következő parancsot:

```
xinit /usr/X11R6/bin/xterm -- :1
```

Figyeljük meg a sor végén lévő `-- :1` karaktereket. Mivel az X már fut a `display :0` kijelzőn, ezt az X-terminált egy másik, jelen esetben az `:1` jelű kijelzőn kell futtatnunk. Egy új virtuális X-folyamat indul el, amely a nyolcas virtuális terminálon fog futni. Elég unalmas a kinézete, mert csak egy egyszerű szürke hátterű X-terminált látunk futni. Nincs sok látnivaló, és nincs mód arra sem, hogy az X-terminál ablakát mozgassuk, de lehetőségünk van parancsok futtatására, sőt más X-programok elindítására is, amelyeket szintén nem fogunk tudni mozgatni. Az új folyamatból a régibe a CTRL-ALT-F7 billentyűkombinációval léphetünk, innen pedig a CTRL-ALT-F8-cal vissza az X-terminálunkba. Egyszerű, non? Ezzel a módszerrel valami érdekesebbet is elindíthatunk, például egy olyan másik ablakkezelőt, mint az IceWM vagy az XFCE, és boldogan kapcsolgathatunk az



2. kép Benépesítjük beágyazott X-kiszolgálónkat

3. kép A KDE beágyazott Gnome-mal, amelybe IceWM-et ágyaztunk

egyik X-folyamatból a másikba, majd vissza. Ha ebből az X-folyamatból ki szeretnénk lépni, ismernünk kell a hagyományos X-window vészkijáratot, amit csak az „ó istenem, mindent megpróbáltam, de nem bírok kilépni az X-ből”-záradéknak nevezek. Egyszerűen nyomjuk meg a CTRL-ALT-BACKSPACE billentyűkombinációt. Ez, mes amis, az X-ből való kilépésnek egy elég durva módja, csak akkor érdemes használni, ha semmilyen más lehetőségünk nem maradt. Az egyik X-folyamatról a másikra való kapcsolás jól működik, de a két asztalon történő munkavégzést eléggé megnehezíti. Az állandó oda-vissza való kapcsolgatás elkerülésére egy Xnest nevű programot használtam. Az Xnest, amely az Xfee86 terjesztési csomag része, érdekes képződmény, mert egy programon belül egyesíti az X-kiszolgálót és ügyfélprogramot. Szó szerint egy beágyazott X-kiszolgálóról van szó. Mielőtt folytatnánk, el kell mondanom, hogy bár a program az X része, előfordulhat, hogy nincs a gépünkre telepítve, jóllehet a csomag (Xfree86-Xnest) nagy valószínűséggel megtalálható Linux-telepítőlemezeink egyikén. A beágyazott kiszolgáló elindításához egy másik `DISPLAY` változót kell biztosítanunk, ahogy az X-terminál esetén is tettük korábban. Mivel a saját X-kiszolgálónk valószínűleg `:0` értékkel fut, válasszuk az `:1`-et. Annak biztosítására, hogy ehhez az új kiszolgálóhoz bármelyik programunkból csatlakozni tudunk, használjuk a `-ac` kapcsolót is. Ez a kapcsoló letiltja a hozzáférés-vezérlést; az & (és) karakter pedig háttérben futó folyamatként indítja el a kiszolgálót:

```
Xnest :1 -ac &
```

Képernyőnkön egy üres ablak jelenik meg, közepén a kiinduló X-kurzorral. Mandrake rendszeremen egy sötétkék színű négyzet volt látható, egy másik kiszolgálón, amin Red Hat futott, ugyanez fekete színű volt. Ezen nem látunk túl sok mindent, indítsunk el hát egy programot az új kiszolgálónkon. Kezdjünk valamilyen egyszerű dologgal, például a mi tiszteletre méltó Xclock programunkkal:

```
xclock -display :1
```

Az ENTER megnyomása után a klasszikus Xclock jelenik meg a második X-kiszolgáló ablakában. Ez remek módja annak is, hogy egy kicsit eljátszadozzunk az X erőforrásaival. Például ezt a meglehetősen unalmas órát öltöztessük fel és helyezzük át a kijelzőnk egy másik részére:

```
xclock -foreground "Red" \
-geometry +450+250 -display :1
```

Ezt az új kiszolgálót egymás után programokkal népesíthetjük be: egy X logó itt, egy Eterm amott. Természetesen ebben a környezetben nem lehetséges az ablakok mozgatása, ezért ennek az ötletnek a határai eddig terjednek. Ha valóban bele szeretnénk kóstolni a munkaasztal-őrületbe, az egész smörg□sbordra szükségünk van, vagyis az ablakkezelőre – és természetesen egy kis borra, hogy könnyítsünk a munka fáradalmain.
Egy teljes ablakkezelő elindítása egyszerű folyamat, a következő lépésként a Tabbed Window Managert (TWM) fogjuk elővenni. Ez a legegyszerűbb ablakkezelő, az Xfree86 részeként juthatunk hozzá. Kezdjük az Xnest programjainak bezárásával, hogy tiszta lappal indulhassunk; ekkor az üres négyzetnek kell előttünk lennie a közepén az X-kur-

zorral. A parancssorba gépeljük be az alábbi parancsot:

```
twm -display :1
```

Ha semmi nem változik, nyomjuk meg a bal oldali egérgombot: erre meg kell jelennie a TWM menüjének. Megmondtam, hogy ez egy nagyon egyszerű ablakkezelő, non? Most próbálkozzunk meg a Window Maker futtatásával:

```
wmaker -display :1
```

Láthatjuk, hogy a formátum lényegében ugyanaz, a `-display` kapcsolót minden esetben át kell adnunk az ablakkezelő indítóparancsának. Itt kell elmondanom, hogy nem minden ablakkezelő használja ugyanazt a kapcsolót. Íme egy lista a népszerűbb ablakkezelőkről, és az indításukhoz használt parancsokról:

- Motif Window Manager: `mwm -display :1`
- F Virtual Window Manager: `fvwm2 -display :1`
- Gnome (figyeljünk a két mínuszjelre): `gnome-session --display :1`
- AfterStep: `afterstep -d :1`

Bizonyos esetekben nem tudjuk az ablakkezelőt a kijelző átirányítását végző kapcsolóval elindítani. Ezek közé tartozik a KDE, az AmiWM, az XFCE és még néhány másik. Ezeknek az ablakkezelőknek a futtatásához először egy egyszerű X-terminált futtassunk beágyazott X-kiszolgálónkon:

```
xterm -display :1
```

Ebből a parancssorból most indítsuk el az ablakkezelőt, parancsként egyszerűen a nevét beírva: `xfce4-session` az XFCE esetében, `amiwm` az AmiWM futtatásához és így tovább. Miután ilyen módon eljátszogattunk néhány ablakkezelővel, egy útakadállyal találhatjuk szembe magunkat. Néha a folyamat közben (különösen a Gnome használata alatt, amikor újabb Gnome-folyamatot akartam indítani) egy hibaüzenetbe botlottam, miszerint a folyamat nem tud elindulni, mert már fut egy folyamatkezelő. Mivel tudtam, hogy nem ez a gond, megnéztem, hogy mire mutat a `SESSION_MANAGER` változó:

```
$ echo $SESSION_MANAGER
local/ultraman:/tmp/.ICE-unix/3132
```

Amint láthatjuk, egy korábbi beágyazott ablakkezelő-futtatás következtében van egy feleslegesen visszamaradt folyamatinformációnk. Az egyik lehetőség a `SESSION_MANAGER` változó értékének a törlése. Egy másik megoldás a zavart okozó fájlok eltávolítása, feltételezve természetesen, hogy ablakkezelőnket már nem futtatjuk:

```
$ rm /tmp/.ICE-unix/3132
rm: remove socket `/tmp/.ICE-unix/3132´? y
```

Úgy tűnik, mes amis, hogy mindjárt itt a záróra, ezért elérkezett a la pi□ce de résistance ideje, vagyis a munkaasztal-desszert, ha jobban tetszik. Néhányatokban felmerülhet a kérdés: ha futtathatok egy beágyazott kiszolgálót, miért ne futtathatnék kettőt vagy hármat is? Egy új beágyazott kiszolgáló elindítása nem jelent többet, mint egy másik display-szám hozzárendelését. A második elindításához írjuk be a `Xnest -ac :2`, a harmadikhoz pedig a `Xnest -ac :3` parancsot és így tovább. Igazából még arra is lehetőségünk van, hogy egy Xnest-et futassunk egy másik Xnest belsejében. A *3. képen* egy KDE munkafolyamaton belül futó Gnome és egy abban futtatott IceWM ablakkezelő látható.
Mint látjátok, mes amis, megvannak az eszközök arra, hogy a legnagyobb munkaasztal-étvágyú felhasználókat is jóllakathassuk. A ***Robert Heinlen***-nek tulajdonított mondás, miszerint „ha ki akarod élvezni az élet minden zamatát, nagyokat kell harapnod”, ebben az esetben azt jelenti, hogy futtassunk minél több különböző ablakkezelőt és munkaasztalt. A záróra most már tényleg elérkezett, de egy kis vizsgálatra még van időnk: François még egyszer újratölti a poharaitokat, mielőtt távoztok. Egészségetekre, a következő viszontlátásig! A vôtre santé! Bon appétit!

Xnest-tipp

Az Xnest egyik leghasznosabb felhasználása, ha grafikus programot fejlesztünk és több ablakkezelővel is ki szeretnénk próbálni művünket. Az Xnest használatával nem kell folyton kilépni, majd az új ablakkezelővel belépni – a segítségével minden ablakkezelővel rövid idő alatt elvégezhetjük a kipróbálást. Így fejlesztési és még inkább kipróbálási időt nyerünk.
Egyszerűen és fájdalommentesen próbálgathatjuk programjainkat, s ha esetleg a beágyazott X-folyamat működésképtelenné válik, akkor egy kattintással kiirthatjuk a rendszerből. Ezután már csak a hibát kell megtalálnunk, hogy a következő próbafutáskor ne történjen meg ugyanez. A lefagyások szinte sohasem vezethetnek oda, hogy a teljes rendszert működésképtelenné tegyék, ennek hasznosságát pedig azt hiszem, nem kell ecsetelnem.

Csontos Gyula

Linux Journal 2004. március, 119. szám

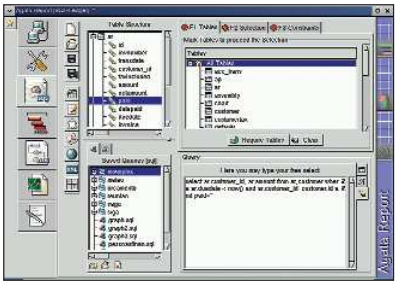
Marcel Gagné (mggagne@salmar.com)
Mississaguában, Ontario államban él.
Ő a szerzője a Kiskapu kiadásában tavaly szeptemberben megjelent Linux-rendszerfelügyelet (ISBN 96-9301-40) című könyvnek.

KAPCSOLÓDÓ CÍMEK

Ablakkezelők az X-hez ➲ http://xwinman.org
Az XFree86 honlapja ➲ http://www.xfree86.org
Marcel honlapja (a borlapot is látogasd meg) ➲ http://www.marcelgagne.com

Agata

Az Agata egy olyan PHP-alkalmazás, amely közvetlenül PHP-ben fut, nem pedig webkiszolgálón. Miért jó ez? Azt nem tudom, de működik. Ráadásul azt mondják, hogyha olyasmire

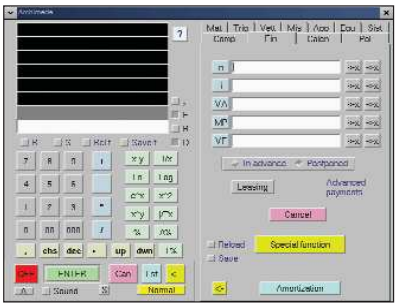


van szüksége az embernek, mint a windowsos Crystal Reports, érdemes egy pillantást vetni az Agatára. A legtöbb SQL-adatbázisra alkalmazható, habár én eddig még csak PostgreSQL-lel vizsgáltam. Az Agata könnyen telepíthető és futtatható, továbbá rendkívül rugalmas. Emellett jól néz ki, és még ábrák készítésére is alkalmas. Működéséhez PHP és php-gtk szükséges.

➲ http://www.agata.org.br

Archimede

Sok évvel ezelőtt, főiskolás koromban volt egy tökéletes programozható számológépem. Amikor legutóbb a kezembe akadt, vastagon borította a por, és nem találtam hozzá elemet, így aztán táblázatkezelőket kezdtem el használni. Most úgy tűnik, hogy újra

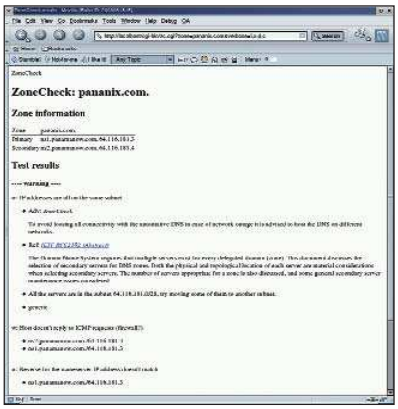


bármikor elővehetem a számológépemet, legalábbis a számítógép képernyőjén. Az Archimede mindenféle függvénnyel rendelkezik, legyen az pénzügyi számítás, trigonometrikus vagy matematikai függvény, sőt naptárszolgáltatást is tartalmaz. Futtatásához libX11, libpthread, libdl és glibc szükséges.

➲ http://mcz.altervista.org

ZoneCheck

Ha nem tudod az összes DNS RFC-szabványt fejben tartani vagy ha a lényeges részeket nem sikerül felidézned, a ZoneCheck segítségével átnézheted a névkiszolgáló zónafájljait, és megállapíthatod, hogy az RFC előírásainak megfelelően tartod-e karban a saját zónádat. Én például megfeledkeztem arról, hogy az időtúllépés számára hétszer nagyobb értéket kell megadni, mint a frissítésnek. Az én



zónámon csak négyszeres volt az időkülönbség, amikor ellenőriztem. Ez a program 105 ehhez hasonló ellenőrzést végez. Futtatható weben keresztül vagy parancssorból is. Előfeltételei: Ruby, a Ruby-kiterjesztések közül a yaml és az exml, illetve egy CGI-támogatással rendelkező webkiszolgáló (ez utóbbi nem kötelező).

➲ http://www.zonecheck

PHP Weather

Egy PHP-parancsfájlról van szó, amelynek segítségével könnyedén jeleníthetjük meg honlapunkon a pillanatnyi időjárási adatokat. Mindezt úgy éri el, hogy időnként lekéri a legutóbbi METAR-adatokat. Ez a METAR-adat elsődlegesen a repülésirányításban használatos; adatokat tartalmaz a hőmérsékletről, szélerősségről, szélirányról stb. Ezeket az adatokat gyűjti be a parancsfájl, majd angolszász vagy épp metrikus mértékegységrendszerre váltja át, s ezután MySQL- vagy Postgres-adatbázisokban átmeneti tárba helyezi (cache). Nekünk nincs más dolgunk, mint az oldalunkon beszúrni egy PHP-kódsort, amely az általunk áhított időjárási adatokat tartalmazza.

➲ http://sourceforge.net/projects/phpweather/

Simple Rescue CD

A csomag katasztrófaelhárító CD-t kínál arra az esetre, ha a rendszerünk valamiért nem lenne képes betöltődni (boot). Ennek számtalan oka lehet: fájlrendszerhiba, fizikai hiba, beállítási hiba, vagy épp csak annyi, hogy a rendszermag újrafordítása után elfelejtettük kiadni a `lilo` parancsot. Az ilyen nehézségek egyszerű megoldására gyógyír az az indítólemez, amely a csomagból készíthető összeállítás CD-re írásával hozható létre. Népszerűségét annak köszönheti, hogy valahol félúton van az alapszintű indító-hajlékonylemez és a nagyméretű, nagy tudású korongok között, ezáltal kellően sok szolgáltatást nyújt, ugyanakkor kis helyen is elfér: például névjegykártya méretű CD-n vagy egy több szakaszos (multi-session) CD végén. A program jelenlegi változata például 7 MB helyet követel magának.

➲ http://www.ivarch.com/programs/rescue.shtml

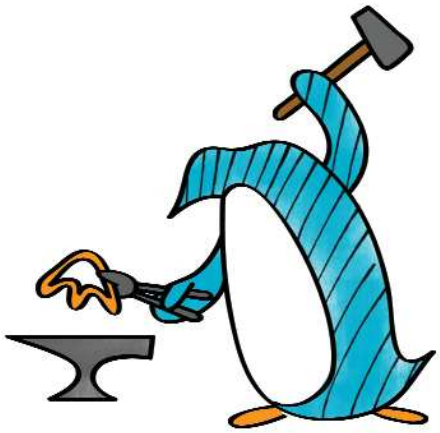
KNetworkconf

Nevéből következően egy, a KDE munkakörnyezethez illeszkedő hálózati beállításokat kezelő alkalmazásról van szó, amely kifejezetten a TCP/IP-jellemzők testreszabását teszi lehetővé. A program folyamatos fejlesztgetés alatt áll, jelenleg a 0.5-ös változatnál tart, ennek következtében a képességei még korlátozottak. Egyelőre csak a már telepített hálózati eszközök beállításait engedi, újat nem lehet felvenni. Emellett természetesen beállíthatjuk az átjárót, a DNS-kiszolgálókat, mi több, még az ismert hálózati helyek (*/etc/hosts*) szerkesztését is lehetővé teszi.

➲ http://knetworkconf.sourceforge.net/

David A. Bandel
(dbandel@pananix.com)
Jelenleg Panamában él, Linux- és Unix-tanácsadással foglalkozik.

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



Bricolage-sablonok

A tartalomkezelő rendszer használata nem feltétlenül jár együtt az egy kaptafára készült oldalakkal. Készítsünk saját sablont!

Az elmúlt néhány hónapban a PostgreSQL-, `mod_perl`- és Apache-alapokra építkező, nyílt forrású tartalomkezelő rendszer (CMS), a Bricolage rejtelmeit boncolgattuk. A Bricolage elég nagy figyelmet kapott az elmúlt években, részben nyílt forrású engedélye, részben az alapját képező nyílt forrású technológiák, részben pedig a kereskedelmi CMS-csomagokkal összemérhető képességkészlete miatt, nem beszélve a nyílt forrású CMS lehetőségeiről.
Ebben a hónapban Bricolage-körutunk utolsó állomásaként a sablonok alkalmazási lehetőségeit ismerhetjük meg. Eddig a Bricolage számos felügyeleti és szerkesztő képességét mutattuk be, de egy CMS csak akkor igazán használható, ha a kimenete testreszabható és vonzóvá tudja tenni az eredményül kapott honlapot.

A sablonok elmélete

HTML-sablonok már elég régen léteznek, ügyes kompromisszumként valahol a statikus oldalak és a HTML programba pakolása között, ahol az a tervező számára elérhetetlenné válik. Amennyiben dinamikusan változó HTML-lapot szeretnénk, nagyon jó választás a sablonok használata.
Természetesen ezzel együtt felvetődik egy fontos kérdés: melyik sablont használjuk? Százszámra léteznek sablonozó rendszerek, ideértve a több tucatnyi Perl nyelven készültet is. Néhány ezek közül, mint a `Text::Template`, önmagában nem is HTML-sablonozó rendszer, mégis nagy sikerrel alkalmazták különféle webalapú feladatokra.
A „Melyik sablont használjuk?” vita talán csak a Linux/BSD és az Emacs/vi elsőségről szóló vitákkal mérhető össze. Szerencsére a Bricolage minden vita fölé emelkedik, ahol békében él együtt a `HTML::Mason` és a Template Toolkit (sablonozó eszközrendszer). Elméletben akár másik sablonozó rendszert is használhatunk, ez a kettő azonban olyan hatékony és népszerű, hogy a legtöbb Bricolage-felhasználót bőségesen kielégíti. Személy szerint jobban kedvelem a `HTML::Mason` rendszerét, ezért a Bricolage sablonkezelését is a Masonön keresztül mutatom be, de ha valaki esetleg Template Toolkit-rajongó lenne, nyugodtan használja azt.
A sablon tulajdonképpen a HTML-oldal tartalmának a vázlata. Minden ide tartozik, amit statikusnak szánunk, a változókat kivéve, amelyek értékét futásidőben adjuk majd meg (interpoláljuk, ahogy programozónyelven mondják). Példaképpen vegyük a következő sablont:

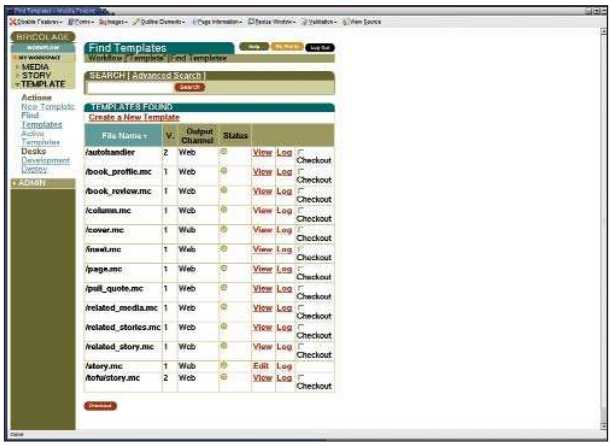
```
<html>
    <head>
        <title><% $title %></title>
    </head>

    <body>
        <h1><% $headline %></h1>
        <p><% $body_text %></p>
    </body>
</html>
```

A fenti, Mason-szabályok szerint készített sablonban mindig van cím (title), címsor (headline) és bekezdés; a `title`, `headline` és `body text` pedig változó.
A Mason két globális változót is rendelkezésünkre bocsát: a `$r`-t, ez a szabványos `mod_perl` objektum Apache kiszolgálónk belső értékeihez enged hozzáférést a Perl API-n keresztül; valamint a `$m`-et, ez az objektum a teljes Mason-környezettel és az adott Mason-sablonnal kapcsolatos további adatokat tartalmaz.
A Bricolage ehhez a keverékhez további három változót ad hozzá. A legfontosabb közülük a `$story`, amely a pillanatnyi történettel kapcsolatos adatokat tárolja. A történetek különféle elemeket tartalmaznak, amelyek maguk is további elemeket foglalhatnak magukba; az éppen pillanatnyi elemet a `$element` változón keresztül érhetjük el. Végül a Bricolage az égető (burner) nevű megoldáson keresztül küldi az oldalakat a kimenetei csatornára (ami általában, de nem szükségszerűen, a weboldal lesz).
Mielőtt folytatnánk, érdemes megismernünk a Mason önműködő kezelőit (autohandler): ezek segítségével egyedi kinézetet kölcsönözhetünk egy oldalnak. Amennyiben egy adott könyvtárhoz létezik autohandler, a fájl vagy könyvtár helyett először mindig ez hívódik meg. Ennek megfelelően, ha a */abc/def.html* és a */abc/autohandler* egyaránt létezik, a */abc/def.html* helyett a */abc/autohandler* fog elindulni. Ez elsőre egy kicsit furcsának tűnhet, és valóban az is lenne, ha nem vennénk figyelembe, hogy az autohandler az `$m->call_next()` hívás segítségével az eredeti sablont bármikor meghívhatja.

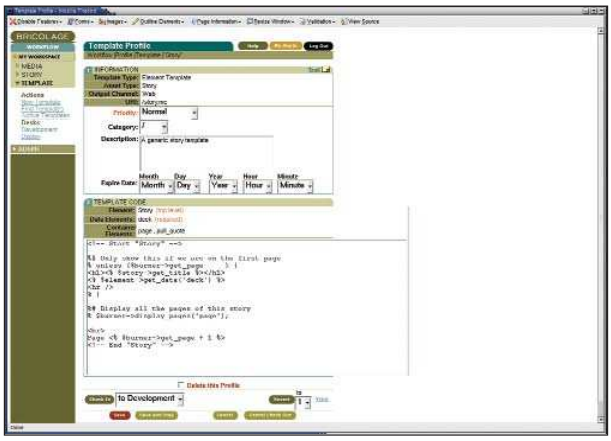
Általánosan használt megoldás, hogy az autohandler belsejébe annyi közös részt préselünk, amennyit csak tudunk, ideértve a menüket, a képeket és a fejléceket. Az autohandler pontosan olyan Mason-sablon, mint a többi, meghívásának különleges körülményeitől eltekintve. Az autohandler belsejében különféle címsorok, képek és menük közt valahová be kell szúrnunk a `$m->call_next()` hívást, amely aztán beilleszti magát a kérelmezett lapot. Ezzel megtartjuk a több sablon használatán alapuló moduláris szerkezet előnyeit, mindeközben egyetlen állomány szerkesztésével lapunkat újratervezhetjük.
Az önműködő kezelők (autohandlers) egymásba ágyazhatók, azaz a Mason az összes fellelhető autohandlert az összes szülőkönyvtárban meg fogja hívni. Tehát a */abc/def.html* hívásakor a Mason először a */autohandler* állományt hívja meg, majd a */abc/autohandler*-t, amit végül a */abc/def.html* követ. Ezáltal szakaszérzékeny (section-specific) kinézetet, illetve szakaszérzékeny (section-based) menüket, alkotórészeket hozhatunk létre.

Sablonok módosítása

Feltételezem, hogy már túl vagyunk az első történet létrehozásán és kiadásán. Amennyiben valaki nem igazán tudja, hogyan kellene történetet létrehozni és kiadni, kukkantson bele az utóbbi néhány számban megjelent Bricolage témájú írásaimba. A történet a kiadását követően valamelyik kimeneti csatornára kerül a fájl helyi másolásával vagy FTP segítségével történő távoli rendszerre juttatásával.
Történetünk kinézetét a sablon határozza meg, de még mielőtt megalkotnánk az első egyszerű sablonunkat, vessünk egy pillantást a rendszerrel együtt érkezett alapsablonokra! Lépjünk a ***Find Template*** hivatkozásra a ***Template*** menüben, a Bricolage rendszergazdai felületének bal oldalán (felhasználói név ***administrator***, a jelszó `change me now!`), majd kattintsunk a ***Search*** gombra. A szövegmezőbe ne írjunk semmit. Most a sablonok listáját kell látnunk, a rendszerben megtalálható minden elemtípushoz egyet-egyet (*1. kép*).
A Bricolage-sablonok több szempontból is hasonlítanak a történetekre: különféle asztalokon hozzuk létre, szerkesztjük és telepítjük őket; elérésük néhány felhasználóra és csoportra korlátozódik; a Bricolage pedig egyszerű változatkezelő rendszere segítségével nyomon követi a változásokat (és megakadályozza az ütközéseket). És valóban, ha megnézzük, az 1. képen azt láthatjuk, hogy valamennyi sablonhoz tartozik egy változatszám. Mindegyiket kikérhetjük a változatkezelő rendszerből, ha rákattintunk a megfelelő jelölőnégyzetre, majd a lap alján található ***Checkout*** (kikérés) gombra. A kikért sablonok az ***Active Templates*** menü ***Templates*** hivatkozás pontja alól érhetőek el.
Bricolage alatt a történet csak egy a terjeszthető elemek közül, továbbá minden elem tetszőleges számú további elemet tartalmazhat. A Bricolage néhány előre meghatározott felső szintű elemet tartalmaz, ilyen például a történet, a könyvkritika és a rovat. Ezenkívül néhány további elemet is, amelyeket más elemekbe ágyazhatunk, ilyen például az idézőjelezés (pull quote).
Ha magunk elé képzelünk egy napilapot, láthatjuk, hogy minden rovat más stílusú, még a hasonló elemeik is. Például a New York Times Metro rovatának hasábjai egészen másképpen festenek, mint a Business rovatéi, és egyik sem hasonlít az Op-ed oldalra (ezeket szaknyelven önálló rovatoknak nevezik). A Bricolage ezt a feladatot úgy oldja meg, hogy az elemeket kategóriába sorolhatjuk. Ha tehát a Sports rovatba írunk, megjelölhetjük, hogy az a ***sports*** kategóriába tartozik. Amikor a Bricolage kihelyezi rovatunkat a webre, a */sports/column.mc* sablon szerint fogja megjeleníteni. Amennyiben létezik, a Bricolage ezt a sablont használja. Ha nem létezik ilyen, kikeresi a legfelső (root) kategória ***column.mc*** sablonját. Más szavakkal, amennyiben egy elemhez felső szintű sablont rendeltünk, azt helyettesítőként vagy alapértelmezettként használhatjuk a rendszer valamennyi ilyen típusú eleméhez. A kategóriafüggő sablonok alkalmazásával lapunk valamennyi szakaszához egyedi kinézetet rendelhetünk.
Mint azt a *2. képen* láthatjuk, kikértem a */story.mc* sablont, amely a történetekhez tartozó felső szintű sablon. Megtekintés (***view***) hivatkozás helyett most már a szerkesztés (***edit***) hivatkozás is elérhető, tehát átszerkeszthetem a



1. kép Válasszuk ki a szerkesztendő elemet a Find Templates lapon!



2. kép Kikért (Checked-Out) sablon szerkesztése

sablont. Olyan módon is megtehetném ezt, hogy belépnék az ***active templates*** oldalra, amelyen hasonló szerkesztőhivatkozást találnék. Nyissuk meg a sablont szerkesztésre, így a 2. képhez hasonló eredményt kell kapunk.
A sablonok szerkesztése igen hasonló a történetek vagy más elemtípusok szerkesztéséhez, eltekintve attól, hogy most épp azt a tárolót módosítjuk, ahol a történetünket később majd elhelyezzük. Amennyiben csak statikus HTML-szerkezeteket

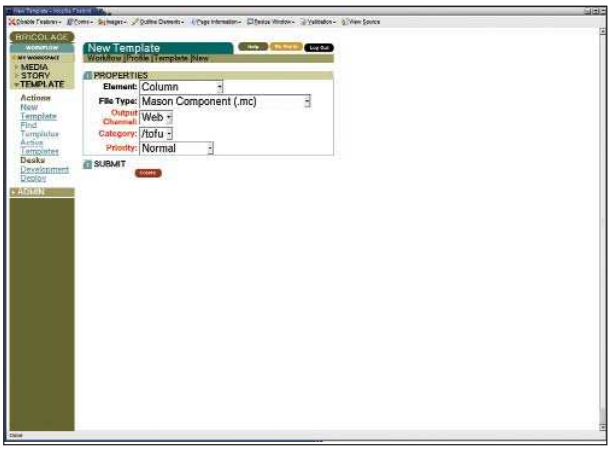
használunk, minden elem pontosan ugyanúgy fog kinézni. Ezért a trükk éppen az, hogy az oldal dinamikus tartalmának létrehozásához felhasználjuk az előre meghatározott `$story`, `$element` és `$burner` objektumokat. Példaképpen álljon itt az alapértelmezett */story.mc* sablon:

```
<!-- Start "Story" -->

%# Only show this if we are on the first page
% unless ($burner->get_page     ) {
<h1><% $story->get_title %></h1>
<% $element->get_data('deck') %>
<hr />
% }

%# Display all the pages of this story
% $burner->display_pages('page');

<br>
Page <% $burner->get_page + 1 %>
<!-- End "Story" -->
```

Láthatjuk, hogy a fenti sablon meglehetősen egyszerű. Egy komolyabb webhely ide különféle elemeket illeszthet be: CSS stílusokat vagy egyéb, az oldalt azonosító statikus szövegeket. A */story.mc* alapváltozata a következőket végzi el:

- A `$burner->get_page()` változóból lekéri a pillanatnyi oldalszámot. A lapszámozás 0-val kezdődik, mi azonban az első oldalon a címet és az elemek listáját jelenítjük meg. A címet a `$story` változóból vesszük a `$story->get_title()` kifejezés segítségével, az adatokat pedig magából az elemből szedjük ki. Figyeljük meg az `$element->get_data()` általános alakú hivatkozást – az elem bármely mezőjéhez hozzáférhetünk az `$element->` meghatározás alkalmazásával.
- A `$burner->display_pages('page')` meghívásával az égetőből kikérve megjelenítjük a történetet.
- Végül ismét felhasználjuk a `$burner->get_page()` utasítást, hogy az oldal alján megjeleníthessük a lapszámot.

Mi történik, ha eltávolítjuk a lapszámot és a helyére a sablon tetején a saját statikus HTML-szakaszunkat szúrjuk be? Változtatásaink a rendszer valamennyi történetén meg fognak látszani. Nem szabad elfelejtenünk, hogy nem minden elem történet, ennek megfelelően a */story.mc* megváltozása nincs hatással a rovatokra, a könyvkritikákra és más elemtípusokra.
Miután elkészültünk a */story.mc* szerkesztésével, rákattinthatunk az oldal alján található ***Check-In*** gombra. Ezzel beolvastuk a sablont, így most már elküldhetjük az (alapértelmezés szerinti) fejlesztőisablon-asztalra (development template desk) vagy azonnal telepíthetjük. Ez az utóbbi lehetőség különösen akkor hasznos, ha a sablonon felfedezünk valamilyen hibát, ami az egész lapra hatna; könnyen módosíthatjuk a sablont, telepíthetjük és azonnal megnézhetjük az eredményt.
Végül megfigyelhetjük, hogy a */story.mc* egyáltalán nem tartalmaz <html> vagy <title> tagokat. Az ok egyszerű, ezeket a részeket az autohandler foglalja magában.
A */autohandler* sablon, amelyet a ***Template*** menüponton



3. kép Kimeneti csatorna és kategória kiválasztása az új sablonunkhoz

keresztül megtekinthetünk, kikérhetünk és szerkeszthetünk, alapértelmezés szerint a következőképpen néz ki:

```
<!-- Start "autohandler" -->
<html>
    <head>
        <title><% $story->get_title %></title>
    </head>
    <body>

    % $burner->chain_next;
    </body>
</html>
<!-- End "autohandler" -->
```

A teljes oldalrendszeren keresztül érvényes autohandler a történet címét a megfelelő HTML <title> tagok közé helyezi, s a `$burner->chain_next()` meghívásával egyúttal a megfelelő laptartalmat is beilleszti.
Ha például általános CSS stíluslapot szeretnénk használni vagy közös menüt akarunk minden oldal tetejére szerkeszteni, esetleg cégünk jelvényét kívánjuk az összes lap tetején megjeleníteni, ebben az autohandlerben tehetjük meg a legkönnyebben. Minthogy az önműködő kezelők egymásba ágyazhatók, a teljes oldalrendszerhez lehet egy általános autohandlerünk, valamint szakaszra jellemző kezelőink az egyes szakaszokhoz.

Sablonok készítése

Mostanáig létező sablonokkal foglalkoztunk, mindazonáltal új sablont létrehozni is éppen ilyen könnyű feladat. Egyszerűen lépjünk a ***Template*** menüpontra és kattintsunk a ***New Template*** hivatkozásra; ott a *3. képhez* hasonlókat kell látnunk. Meg kell adnunk a sablonunkhoz tartozó kimeneti csatornát és a kategóriát. Kattintsunk a ***Next***-re, majd megadhatjuk, hogy melyik kategóriára vonatkozzon a sablonunk.
A kategória-csatorna-elem együttesnek egyedinek kell lennie, ennek megfelelően egy csatornához, kategóriához vagy elemhez több sablont is rendelhetünk. De a ***Web output channel in the root (/)*** kategóriájának ***story*** eleméhez csak egyetlen sablon tartozhat. Ha megpróbáljuk megszegni ezt a szabályt, a Bricolage figyelmeztet bennünket,

hogy erre a kombinációra már létezik sablon. E nehézségre több megoldás is létezik; az egyik, hogy egyszerűen új elemtípust hozunk létre, a másik, hogy új kategóriát készítünk, végül akár az adott kombinációhoz tartozó eredeti sablont is megváltoztathatjuk. A legjobb megoldás mindig az adott elérendő céltól függ.
Most készítsünk új sablont a *tofu* kategória rovataihoz, amelyet Bricolage alatt a */tofu/column.mc* állomány képvisel. Miután rákattintok a *Create* gombra, egy szerkesztőoldalt kapok, ahol létrehozhatom vagy módosíthatom a sablonomat. Én a sablonomat rendkívül egyszerűre tervezem:

```
<!-- Start "tofu/column" -->

%# Display this story
% $burner->display_pages(´page´);

<!-- End "tofu/column" -->
```

Figyeljük meg, hogy a meghatározás köré HTML-megjegyzéseket helyeztünk. Ezáltal könnyebb lesz a sablonjainkban megkeresni a hibát, miután az HTML alakban a felhasználó böngészőjére került. A saját tapasztalataim alapján bátran kijelenthetem, hogy a Mason egymásba ágyazott sablonjai, különösen többszörös önműködő kezelőhasználattal, igencsak őrjítőek tudnak lenni.
Miután a *Check-In* menüből kiválasztom a telepítést (deploy), majd rákattintottam a *Check-In* gombra, sablonom beépült a rendszerbe. Ettől kezdve bármely rovat, amely a *tofu* kategóriába tartozik, az általános sablon helyett az új sablonunkat használja.
Természetesen, ha vissza szeretnék lépni, hogy átszerkesszük a sablont, a korábban látott módon nyugodtan megtehetjük – kikeressük, kikérjük, majd átszerkesztjük.

Összegzés

A Bricolage, mint minden komoly CMS, sablonok segítségével teszi egyszerűvé az egységes oldalkép kialakítását. Minthogy a Bricolage olyan általános nyílt forrású eszközökön alapul, mint a `mod_perl` és az Apache, kihasználhatja a `mod_perl`-ben már létező sablonozás előnyeit, ideértve a `HTML::Mason` és a Template Toolkit-megoldásokat. Ebben a hónapban megnéztük, hogyan hozhatunk létre és módosíthatunk különféle elemtípusokhoz és kategóriákhoz tartozó sablonokat, ilyen módon kötöttségek nélkül juthatunk hozzá a honlapunk kinézetének egységesítését biztosító rugalmassághoz.
A cikkhez tartozó Kapcsolódó címek megtalálhatóak a ➲ http://melleklet/linuxvilag.hu/Bricolage könyvtárában.

Linux Journal 2004. március, 119. szám

Reuven M. Lerner (➲ http://www.lerner.co.il/atf) Nyílt forrású programokra, valamint web- és adatbázis-alkalmazásokra szakosodott tanácsadó. Könyve, a Core Perl, 2002 januárjában jelent meg a Prentice Hall gondozásában. Reuven feleségével és lányaival nemrég költözött Chicagóba.

Az IBM eServer BladeCenter

Az IBM az utóbbi időben x86 alapú Linux-rendszereit két nagy kiszolgálócsaládba csoportosította. A méretezhető rendszereknél az IBM az eServer x440 sorozatot ajánlja, amely egyetlen gépben 16 processzort képes kezelni.

Az átlagos felhasználó számára az IBM eServer BladeCenter rendszere az érdekesebb, amelyben egy keretbe szerelhető, 7U magas házban 14 pengét használhatunk egyszerre. Maga a ház csupán egy passzív középlemezből, egy CD-ROM-olvasóból, hajlékonylemezes meghajtóból és két 25 cm-es ventilátorból áll. Két redundáns 230 V-os tápegységet találunk benne, de további kettőre is szükség lesz, amennyiben egyidejűleg ötnél több pengét szeretnénk használni. Egy billentyűzetből, videokártyából, egérből, illetve ethernetcsatlakozókból álló kezelőegység (KVM) is tartozik az egységhez, s az új modellek már a redundáns kezelőegységeket is támogatják.

A redundancia a BladeCenter egyik fő tervezési irányelve, így szinte mindent ennek alárendelve építettek meg. A ház hátulján négy helyet látunk a kapcsolómodulok (switchek) számára. A második hely az első hellyel, a negyedik hely a harmadik hellyel áll redundáns kapcsolatban. Ha például az első helyre egy ethernetkapcsolót telepítünk, akkor a harmadik helyre egy másodikat is telepíthetünk, de nem SAN-típusút. Bár ez így teljes redundanciát jelent, a fentiek alapján egyszerre csupán kétféle kapcsolómodult helyezhetünk el a házban.

Jelenleg az IBM gigabites ethernet- és szálcsatornás SAN kapcsolómodulokat kínál. Ezek a kapcsolómodulok belül minden egyes pengéhez csatlakoznak, kívül pedig a gigabites kapcsoló négy, rézből készült ethernetkapuval bír, míg a SAN kapcsolómodulnál négy SAN-kaput használhatunk. A redundáns modulok telepítése után minden penge két ethernet- és két SAN-kaput ér el. Az ethernetkapcsoló egy minden szolgáltatást magában foglaló hármas rétegű kapcsoló, amely hatékony webes felülettel rendelkezik. A Q-logic típusú SAN-kapcsolót `telnet` kapcsolaton keresztül vagy a Linux és Microsoft Windows alá is létező, Java alapú kezelőfelület segítségével állíthatjuk be. A BladeCenter házának elülső oldalán egy CD-olvasó, hajlékonylemezes meghajtó és USB-csatlakozó található – ezt az utóbbit egy belső USB csomópont köti össze az egyes pengékkel. Mivel egyszerre csak egy penge használhatja a CD-t, a hajlékonylemezt és az USB-csatlakozót, az USB üzembe helyezéséhez a megfelelő pengén meg kell nyomnunk egy gombot. A billentyűzet, a videokártya és az egér egy újabb belső csomóponton keresztül csatlakozik a pengékhez. A vezérelni kívánt penge kiválasztásához egy minden pengén megtalálható gombot kell megnyomnunk.

A KVM és az USB-eszközöket a BladeCenter webes kezelőfelületével is kezelhetjük. Bár további (redundáns) KVM-kezelőegységeket is csatlakoztathatunk, sajnos a konzolhoz még a webes felület segítségével sem rendelhetünk egynél több pengét. Ez valószínűleg a BladeCenter legbosszantóbb hiányossága, de talán kisebb gondot okoz azoknak, akik a rendszert egy valódi kiszolgálótelep részeként használják.

E cikk írásakor csak a HS20 kiszolgálópenge létezett a BladeCenterhez. Könnyen azt hihetnénk, hogy e pengék mindegyike egy módosított ThinkPad egyetlen lapra zsúfolva, azonban minden penge összezsugorított vállalati osztályú kiszolgáló. Bár egyetlen processzorral és 512 MB memóriával szállítják, a HS20 533 MHz-es FSB-vel rendelkezik, emellett két, legfeljebb 2,8 MHz-es Pentium 4 Xeon processzor és négy PC2100 típusú DDR DIMM (összesen legfeljebb 8 GB) memória számára van rajta hely.

A Xeon CPU hiperszálas tulajdonságainak köszönhetően, a mai Linux-magok két logikai processzorként látnak minden processzort.

Minden penge rendelkezik egy 8 MB-os Rage képvezérlővel és két BroadCom ethernetvezérlővel. Mindkét ethernetvezérlő felületszerelten csatlakozik egy kapcsolóegységhez, a meghajtóprogramban pedig beállíthatjuk, hogy az ethernetvezérlőket egymástól függetlenül, vagy az egyiket a másik tartalékaként szeretnénk használni.

A pengéhez két notebook méretű IDE-merevlemez csatlakozhat, de a beépíthető SAN-vezérlő a második IDE-meghajtóval nem fér össze, így a SAN használatakor csak egy merevlemez lehet a pengén. A SAN-vezérlő egy egyedi Qlogic SAN kártya, amely

Termékismertető

Gyártó: IBM
Ár: BladeCenter ház 2789 dollártól, HS20 penge 1879 dollártól, ethernet kapcsolómodul 2199 dollártól.
➲ http://www.ibm.com/servers/eserver/bladecenter

redundáns útvonalakat biztosít a SAN kapcsolómodulok felé. Az IBM egy kiegészítő pengét is kínál, mely minden pengéhez két SCSI meghajtó csatlakoztatását teszi lehetővé, de ez is lefoglal egy bővítőhelyet. A pengék üzem közben cserélhetők, a frissítésük egyszerű. Ha egy penge végzetes hiba áldozatául esik, akkor gond nélkül eltávolíthatjuk, a figyelőrendszer pedig a hibás alkatrész mellett villogó lámpával jelzi a problémát. Úgy vettem észre, hogyha a második processzorra hűtőbordát szerelünk fel, akkor a rugós csavarok miatt némi erőszakot is szükséges alkalmaznunk.

A BladeCenter másik érdekes sajátossága, hogy különböző – Intel és nagyszámítógépes – pengéket egyaránt használhatunk a házban, akár egyidejűleg is. Az IBM ígérete szerint hamarosan megjelennek hozzá a pSeries és iSeries típusú kiszolgálóikra épülő pengék, e cikk írásakor azonban még egyetlen ilyen terméket sem jelentettek be hivatalosan.

Mint már korábban is említettem, a BladeCenter hatékony webes felülettel rendelkezik, amelynek segítségével kibekapcsolhatjuk a pengéket; továbbá a CD- és hajlékonylemez-meghajtó csatlakozóját vagy a konzol kapuját kapcsolhatjuk át, illetve a gép firmware-jét frissíthetjük weben keresztül. Jelenleg a felhasználói neveket és jelszavakat a kezelőkonzolon kell beállítanunk, ami bosszantó, hiszen ezeket a felhasználóneveket és jelszavakat minden ethernet és SAN kapcsolómodulban újra be kell állítanunk.

A biztonsági beállítások is kezdetlegesek: a felhasználók nem kaphatnak hozzáférést, csak bizonyos pengékhez, illetve nem korlátozhatjuk a tevékenységüket bizonyos feladatokra. Bár lehetőségünk adódik egy, csak az adatok megtekintésére szolgáló hozzáférés létrehozására, így azonban a konzol lehetőségeit sem használhatják.

A webes felület mellett a BladeCenter az igen hatékony IBM Director 4.1-es változatának felhasználási jogát is tartalmazza, amely olyannyira sokrétű, hogy az ismertetése külön cikket érdemelne. Fontos megemlítenünk, hogy a program ügyfél-, ügynök- és kiszolgálórésze egyaránt fut Linux alatt és a rendszer működésének elemzésére, folyamatok indítására és leállítására használhatjuk. Emellett lehetőségünk nyílik arra, hogy egyetlen konzolon tekinthessük meg az összes IBM kiszolgálóval kapcsolatos jelentést.

A BladeCenter Linux támogatása sokkal jobb, mint bármely más cég gyártotta rendszeré. A csomagban megtaláljuk a Linux-meghajtókat tartalmazó CD-ket és minden megkapott program Javában íródott, teljes Linux-támogatással. Igazság szerint még nem találkoztam olyan vezérlő-, figyelő- vagy beállítóprogrammal, amelyből csak Windows-változat létezik. Ez üdítő változás a korábbi időkhöz képest, hiszen efféle gondok még a Linux leghűségesebb támogatóiként számon tartott gyártók esetében is előfordultak. Bár a rendszer jelenleg hivatalosan csak a Red Hat Linux 7.3-at, a Red Hat Advanced Server 2.1-et és a SuSE Linux 8-as telepítéseket támogatja, nálam gond nélkül működött Red Hat Linux 8.0-s és 9.0-s alatt is.

Aki egyedi alkatrészek, kártyák vagy csatolók használatát tervezi, annak nem célszerű a BladeCentert választania, hiszen itt sem PCI, sem pedig pengénkénti be- és kimeneti csatlakozók nincsenek. Ha minden pengét elláttak volna egy-egy USB-csatlakozóval, akkor jóval sokoldalúbb rendszerről beszélhetnénk.

Egy másik megfontolandó tény, hogyha nincs külön e célra kialakított kiszolgálótermünk, akkor a rendszer magas zajszintje valószínűleg gondot fog okozni. A gép valójában olyan hangos, hogy az IBM külön hangcsillapítót (Acoustic Attenuation Unit, AAU) is árul az eszközhöz, amely lényegét tekintve egy jókora habszivacs szigetelés. A gép hangja még az AAU felszerelésekor is áthallatszik egy iroda falán.

Az IBM állítása szerint a BladeCenter árait úgy alakították ki, hogy hét vagy több penge vásárlása esetén a rendszer a hasonló teljesítményű önálló kiszolgálókhoz képest olcsóbb megoldást kínáljon. Meglehetősen ritkán találkozhatunk olyan önálló kiszolgálóval, amely a BladeCenterhez hasonlóan magas szintű redundanciával rendelkezne, de véleményem szerint az IBM állítása igaz, még a kevesebb kedvező tulajdonsággal rendelkező kiszolgálókhoz hasonlítva is. Összegzésképpen elmondhatjuk, hogy a BladeCenter leginkább a sűrűn telepakolt kiszolgálófürtök építőit fogja sikerrel csábítgatni.

Előnyök

- 14 kiszolgáló és 28 processzor egy 7U magas állványban.
- Teljesen redundáns felépítés.
- Teljes körű, a Windowséval egyenértékű Linux-támogatás.
- Ára hét penge fölött versenyképes az önálló kiszolgálók áraival.

Hátrányok

- Egyetlen konzol 14 rendszerhez.
- A támogatás csak a Red Hat és a SuSE bizonyos változataira vonatkozik.
- Korlátozott bővíthetőség.

Linux Journal 2004, 117. szám

Dana Canfield (canfield@peng1.net)
Az Indianapolisi Egyetem rendszergazdája, innen ösztönzi a gyártókat arra, hogy összes létező terméküket lássák el megfelelő Linux-támogatással.

Nemsokára Unreal Tournament 2004

Az Unreal-sorozat hamarosan új taggal bővül, ugyanis már úton van az Unreal Tournament 2004, amely újabb része e nagy sikerű sorozatnak. Igaz, a jelenlegi leírás csupán ízelítő az előzetesen kiadott próbaváltozatból, de talán sokan kíváncsiak rá, hogy mit is várhatunk tőle.

Régóta nagy Unreal-rajongó vagyok. Még az első Unreal volt az, ami olyan mélyen megragadott, hogy a mai napig érzem a hatását. Ekkor hagytam el a Quake-tábort, és pártoltam át a versenytársához. Bár a Quake II nagyon jól eltalált játék volt, az akkor frissen megjelent Unreal hangulata mellett esélye sem maradt nálam. Ekkor találtam ki az „Unreal feeling" kifejezést, amely örökös etalon lett számomra, és ami mindig alapvető szempont, ha egy játékot értékelek. Egy játék legnagyobb játékélményért felelős tényezője a hangulata, és véleményem szerint ez az, ami sikeressé tesz egy játékot, nem pedig a marketing, ahogyan azt néhány játékgyártó cégnél gondolják.

A csomag

A játék csomagja először a neves magyar játékoldalon, a ➲ http://www.hcgamer.hu címen jelent meg. Jelenleg már az ➲ ftp.linuxfórum.hu weboldal ***games*** részéből is letölthető. A csomag mérete „könnyed" 250 MB körüli, tehát a széles sávú elérés kötelező a letöltéséhez. Meglepetés akkor éri az embert, miután letölti ezt a ***.bz2*** állományt és kicsomagolja, ugyanis egy közel 420 megabájtos futtatható és telepíthető ***.run*** fájl jön létre.
A futtatása teljesen egyértelmű, rendszergazdaként kell elindítani a telepítést, amelynek során az Unreal Tournament 2003-ból már megszokott telepítővel találkozunk. Érdekes, hogy a 420 megabájtos csomag ellenére a telepítés elég gyorsan lezajlik, tehát hamar elmélyülhetünk a játékban.
A telepített program sem sokkal nagyobb, mint a telepítőcsomagja, azonban tegyük hozzá, hogy ezt a kijelentést a 60 gigabájtos merevlemezek korában tettem. Valószínűnek tartom, hogy a végleges változat követi az Unreal Tournament 2003 telepítési elvét, és a linuxos változat telepítése során 4 gigabájtos játékmérettel szinte biztosan találkozhatunk majd.

Játék versus játékélmény

A telepítést követő indítás során az előző változathoz képest valamivel hangulatosabb nyitóképernyővel találkozhatunk; kellemes zene hangzik fel, amely viszont még mindig messze elmarad a Tournament és az első Unreal hangulatától. A játékmódok azonban biztatóbbnak ígérkeznek: az élethalálharc (standard deathmatch), az Assault, a bombing-run mind megtalálható és kipróbálható. A menürendszer átgondolt, könnyen kezelhető, és érezhető rajta az első Tournament néhány eleme, de a 2003 beállítási rendszere, valamint összetevői a jellemzőek.
A jónak gondolt beállítások után először a Deathmatch módot próbáltam ki. A helyszín egy lerobbant gyártelep, amely sajnos egy kicsit sötétre sikeredett. A fegyverek azonban ugyanazok, mint amit már a 2003-ban is megszokhattunk, a botok csak némi átdolgozáson mentek keresztül, és a játékmenet alig lett pörgősebb. A hangulata viszont sokkal jobb lett. Bár a beszólások többsége már ismert lehet a sorozat rajongói számára, sőt a modellek

némelyike is visszaköszön az előző részekből, valamiért mégis hangulatosabbnak érzem a játékot.
Az Assault üzemmód érdekes meglepetést tartogatott. Először is ekkora nyílt tereppel talán még sosem találkozhattunk az Unreal-sorozatban; másodsorban megjelent egy újdonság, amelynek a létrehozására már akadtak próbálkozások, de ennyire jól talán sosem működött még – ez pedig a járművek használata volt. Ezzel az ötlettel én személy szerint először a Jedi Knight II: Jedi Outcast játékban találkoztam, amelyben lehetőség adódott egy birodalmi lépegető vezetésére a Yavin IV-en. Aztán a HALO-ban szinte alapértelmezett lett, és egyszer

csak azzal szembesültünk, hogy a járművek használata egyre inkább nem másodlagos extra lett, hanem elsődleges, hasznos eleme a játékoknak. Így van ez az Unreal Tournament 2004-ben is. A járműnek fegyverei vannak, mellyel a kijelölt célpont megsemmisíthető. Ez nagy előrelépés, és – valljuk be – nem utolsó élménynövelő tényező. A játékos némi FPS után átnyergel egy kicsit a szimulátorok világába.

Motor és gépigény

Amennyire megfigyeltem, a grafika tekintetében szinte semmiféle változás nem fedezhető fel a 2003 és a 2004 között. Nyilvánvalóan a 2003 motorja dübörög a játékban. Némi modellrészletesség-növelés megfigyelhető és valamivel kidolgozottabb is lett, habár könnyen tévedhetünk a pályakinézet – amúgy szerintem semmivel sem fantáziadúsabb, mint a 2003-ban – csalóka volta miatt. Ez a kevéske „grafikai tuning" viszont egyértelműen tetten érhető a gépigényen is. Az én gépemen (800-as Duron, 512 DDR memória, GF2 videokártya) a 2003 még játszható volt, a 2004-ben viszont csak a zárt terekkel bíró pályák futottak használható sebességgel 800×600-ban. Aki nagy játékos, az tudja, hogy a 800×600 már nem a legfelemelőbb felbontás, valamint azt is, hogyha a játék szaggat, garantált a percenkénti respawn. Nos, a 2004 erősen, kiváltképp a bemutatópéldányban indítható hatalmas Assault pályán. Ez majdhogynem teljesen játszhatatlan a fentebb leírt gépen. A végső változat minden bizonnyal már némiképp egyszerűsítve lesz, de nem hiszem, hogy szemmel látható eredményt kapunk. Ha hozzáveszem, hogy a nálam használt Linux-tényező (a Linux-használat esetén fellépő 5–10%-os sebességtöbblet) esetén is ez az eredmény, nos, akkor nagyon erős gép szükséges a játék alá. Nekem az a meglátásom, hogy egy Athlon processzor és egy GF4 javasolt neki, illetve az 512 DDR memória szintén nagyon elkél hozzá. A 2004 – követve az Unreal Tournament1 és a 2003 által kitaposott utat – teljesen a hálózati játékra van kihegyezve. A hálózati játékot sajnos még nem volt lehetőségem kipróbálni, de gyanítom, hogy (kiindulva az előző változatokból) a 384/64-es ADSL kapcsolat tényleg csak belépő szintű sebesség.

Vélemény

Akinek tetszett az Unreal Tournament 2003 és szereti is, az minden bizonnyal jobb játékot fog kapni, mint az előző. A hangulata és a játékmenete egyaránt sokkal jobb. Aki viszont úgy gondolkodik, mint én, tehát az Unreal Tournament1 volt az utolsó olyan Unreal, amelynek megvolt a remek „Unreal feelingje", és nem tetszett neki a 2003, az valószínűleg ismét úgy fog érezni, mint én. Ugyan sokkal jobb, mint a 2003, de megvárja a végső változatot és megnézi a bemutatópéldányt, azaz kivárásos álláspontra helyezkedik, mert bár nem sokban különbözhet a végsőtől, mégiscsak próbaváltozat. A véleményem az, hogy hangulatban és játékélményben még mindig nem sikerült elérni a „régi" Unreal Tournament szintjét, grafikában felülmúlni a 2003-at, a zenei aláfestés pedig fényévekre van az UT1-től (hát még az Unrealtól!), de a játékmenet csiszolására tett erőfeszítések hasznosak voltak.

Dancsok „strogg" Zoltán
(strogg@mail.tvnet.hu)
Jelenleg technikai szerkesztőként dolgozik a BME-OMIKK-nál, ahol oktat is.