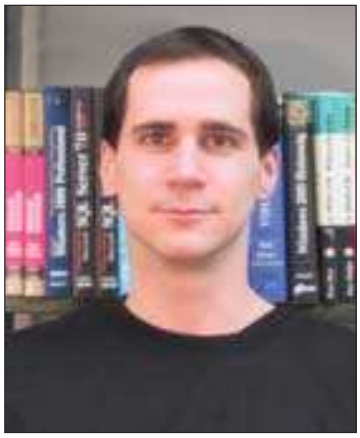
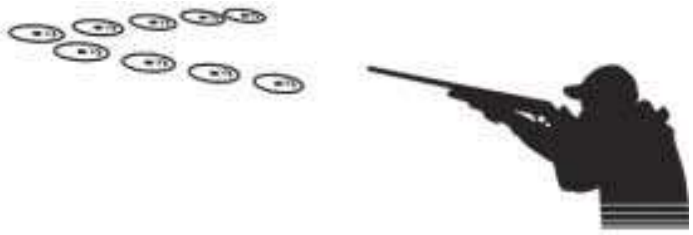
Beköszöntő



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

Webkamerák, héjprogramok, bakelit, költözés, összeomlás. Ahogy először átfutottam a nyomdába készülő anyagot, ezek a szavak maradtak meg bennem. Pszichológusok gyakran használják a szó-memóriatesztet ismeretlen ügyfeleknél: elsorolnak harminc kulcsszót, és megkérik a kezeltet, hogy mondja vissza, ami megmaradt benne. Ha a vér, gyilkosság, kés, kín szavakat mondja vissza a kezelt, egy jól képzett pszichológus elnézést kér, hogy ki kell mennie az illemhelyre, és csak két biztonsági őr kíséretében merészkedik vissza. Vajon én milyen gyorsértékelést kapnék?
Azért megmaradt még egy-két gondolat. Például a Közönségdíjakat (13–15. oldal) olvasgattam, amikor szembetűnt, hogy a Debian az LJ olvasói között a kedvenc Linux-változattá lépett elő. A lapszámban több SuSE-cikk is található (Viktor a 60–63. oldalon kiszolgálónak telepíti, Zoli a 56–59. oldalon kezdők számára tart bevezetést), míg más változatok alig-alig bukkannak fel. Hogy egy kicsit színesítsünk a palettán, elindítottam egy új sorozatot, Debian otthonra címmel (64–65. oldal), amiben két célom van: az egyik, hogy ezt a rendszert is egy kicsit jobban bemutassuk, a másik pedig, hogy a rengeteg szabadidővel rendelkező olvasóinkat rávegyem egy kis türelemjátékra. Ugyanis mi másnak nevezheti az ember egy Debian telepítését? Mellesleg remek tapasztalatszerzés is, sőt munka közben filozófiai magasságokba emelkedve elmélkedhetünk az emberi tűréshatárról. Emellett a mai világban már nemcsak azt kérdezik meg egy új rendszergazdától, hogy ért-e a GNU/Linux rendszerekhez, hanem hogy melyekhez. Hmm. Igaz, hallottam olyat is, amikor az IT-menidzsör viszontkérdése így hangzott: „Linux? Fut az a 2000 Serveren?”
Azért egy kicsit szomorkás vagyok, hogy a Közönségdíjakban a böngészők között nem az Opera van az első helyen. Na jó, de legalább dobogós lenne! Szerintem remek program, még a reklámcsíkjával együtt is. A múltkor egy egész délutánt eljátszottam, hogy megjelenítse a flash-t, és hogy megismerkedjek a beépített levelezőjével, az M2-vel. Igen, játék és szórakozás! Természetesen ha valaki látványosabb játékokra vágyik, akkor ott a Frozen Bubble (első helyezett), vagy a 72–74. oldalon megtalálhatjuk a versenyautókat is, Marcel pilóta vezetésével. Ha valakinek ez sem elég, készítsen magának játékot. Ebben Fábián Zoli segíti a 47–49. oldalon.
És hogy az ember a setét nagyvárosban is tudjon játszani (ahogy erre az MGE óriásplakát-kampánya is felkészít bennünket), de még akkor is, amikor a szomszéd az idegösszeomlás szélén dühében belevágja az asztalilámpát a fürdőkádba, érdemes beüzemelnünk egy szünetmentes tápot (lásd Kolcza Péter cikkét a 80–81. oldalon). Igaz, ma már egy komolyabb kiszolgáló elképzelhetetlen szünetmentes áramforrás nélkül. Szépen is nézne ki mondjuk a SARS genetikai kódját számító linuxos telep, ha a sarokban ott csücsülne az amerikai baseball-válogatott, és tekernék a bicajt. Remélem, hogy mindenki talál érdekes cikket e hónapban is, és sikerült egy színes lapszámot összeállítanunk.
Jó olvasást kívánok!

Programvadászat



Ehavi CD-mellékletünkön a Gentoo Linux 1.4-es első korongját adjuk közre. Ez nem egy mindennapi, csilivili Linux-terjesztés, hanem jól használható és testreszabható rendszer, viszont ennek a célnak az elérésért meg kell egy kicsit szenvednünk. Azoknak, akik könnyen, grafikusan telepíthető Linuxra vágynak, nem valószínű, hogy tetszeni fog (bár láttam már olyan kezdő linuxost a barátaim között, akik a Red Hat telepítőjével nem tudtak mit kezdeni, idegesítette őket, a Debianéra viszont azt mondták, hogy „mindig ilyenre vágytam!" – jó, tudom, ez extrém eset, de létezik ilyen is). Összességében azonban egy nagyon jó és gyors rendszert kaphatunk.

A Gentoo Linux telepítése

A rendszer a CD-ROM-ról indítható. A Gentoo Linuxot többféleképpen telepíthetjük; az egyik lehetőség: gyorsan, előre elkészített és lefordított csomagokkal telepíteni; a másik: ha teljesen testreszabott Linuxot szeretnénk, akkor forrásból telepítsük. Az első CD, amit most adunk közre, csak az előbbi telepítést támogatja, a programok forrásai majd azon a második CD-n lesznek megtalálhatóak, amelyet olvasóinknak a jövő hónapban mellékelünk a lap mellé. Minimum Pentium kategóriás gépet használjunk, jó sok memóriával (a 64 MB memóriával futó gépeken rettentően lassú lehet a programok fordítása).

A telepítő CD-k

Az első korong, amit az olvasó most a kezében tart, az indítható ***LIVE CD Installation***. Ezen a korongon minden megtalálható ahhoz, hogy a Gentoo Linuxot telepíthessük és működő merevlemezről futó Linux operációs rendszerünk legyen.
Erről gyorsan telepíthetjük az előre elkészített csomagokat, nem szükséges végigvárnunk a programok forráskódból történő lefordítását. Megtaláljuk például az Xfree86 kiszolgálót az előre elkészített csomagok között.
A második korong nem alkalmas rendszerindításra, viszont nagyon sok előre elkészített csomagot tartalmaz (KDE, Gnome, OpenOffice.org, Mozilla stb.).

Ez a CD szintén azoknak az eszköztárához adódik, akik gyorsan szeretnék a rendszerüket „talpon" látni. Ezeknek a csomagoknak a lefordítása körülbelül 36 órát venne igénybe egy mostani átlagos számítógépen, úgyhogy sok-sok türelem kellene hozzá, ha forrásból telepítenénk.
A telepítés a hőskorban megszokott módon zajlik: a rendszert elindítjuk a CD-ről, megkapjuk a rendszergazdai parancssorjelet (root prompt), az `fdisk`-kel és a többi segédprogrammal előkészítjük a merevlemezünket, a ***stage*** könyvtárból pedig kicsomagoljuk a megfelelő készletet:

stage1: ez a legkisebb méretű; hálózati kapcsolat szükséges;
stage2: szintén szükséges hálózati kapcsolat;
stage3: nem kell hálózati kapcsolat, önállóan, a CD-ről is telepíthető.

A következő számunkban részletes telepítési útmutatót készítünk!
Addig is: ➲ http://www.gentoo.org

Magazin

A magazinban megjelenő cikkekhez szorosan kapcsolódó, esetleg azoknak szerves részét képező anyagok természetesen szintén helyet kaptak a korongon. A legnagyobb terjedelmű anyagok ***Marcel Gagné*** cikkéhez kapcsolódnak.
A bemutatott autóversenyző programokat a legkülönfélébb Linux-változatokhoz előre csomagolt formában vagy forrásként találhatjuk meg a korongon – ha netán egyik sem futna a gépünkön (Magazin/Fogadó könyvtárban).
A bakelitlemezeiket féltve őrző és sajgó szívvel hallgató olvasóinknak a Magazin/Bakelit könyvtárban nyújtunk segítséget egy esetleges CD-másolat készítéséhez, a cikkben bemutatott programok közreadásával.
Aki otthon vagy a munkahelyén, netán az interneten található kiszolgálóján szeretne rádióadót működtetni, annak figyelmébe ajánlom a Magazin/SHOUTcast könyvtárat – az ebben fellehető programok és a cikk segítségével egy kis bütykölés után máris üzembe állíthatjuk a kiszolgálónkat.

Rendszermag

A legújabb fejlesztői mag a 2.6.0-pre9-es nevet viseli, és a rendszermagfejlesztők mindent megtesznek, hogy megbízható rendszermag kerüljön a kezeink közé: egy csomó új dolog kerül bele, és teljesen újraírt rész cserélődik ki a régivel.
A magazinban diff -u, illetve Rendszermag-fejlesztési hírek címen olvashatunk a legújabb fejleményekről.

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

DriverLoader

A Linuxant új termékének köszönhetően már nemcsak Windows operációs rendszereket és az ezek alá készült alkalmazásokat lehet futtatni Linux alatt, de windowsos illesztőprogramokat is. A DriverLoader 1.2-es változata egy olyan újfajta burkoló, amely a szabványos Windows NDIS, vagyis a hálózati csatolókhoz készített illesztőprogramok futtatását teszi lehetővé x86 alapú gépen futó linuxos rendszeren. Az illesztőprogramokon semmilyen módosítást nem kell végezni. A DriverLoader elsősorban azoknak a felhasználóknak az életét könnyítheti meg, akik valamiért nem találnak megfelelő linuxos illesztőprogramot egy adott eszközhöz, de az illesztőprogramok fejlesztését túl nagy tehernek érző gyártók számára is új lehetőségeket kínál. A DriverLoader forrása nyílt, így tetszőleges rendszermag alatt megoldható a futtatása. A DriverLoadert 30 napig bárki ingyenesen próbálgathatja, utána azonban használatának jogáért 20 dollárt kell fizetni.

➲ http://www.linuxant.com

Bővülő irodai választék

Az 1.1-es OpenOffice után a Sun által összeállított StarOffice 7-es változata is elkészült. Az irodai programcsomagot a Sun – immár hagyományosan – Linux, Solaris és Windows operációs rendszerek alá készítette el. Az újdonságok listája a lényegében azonos alkalmazások miatt a nyílt csomag kiadásakor megismerttel egyezik, a leglényegesebb eltérést az árban fedezhetjük fel.

A Sun webhelyéről letöltve a StarOffice 80 dollárba kerül, ugyanezt a műveletet a Lindows.com oldaláról 60 dollárért indíthatjuk el. A nagyobb tételben vásárló vállalatok akár 25 dollárra is leszoríthatják a gépenkénti árat, ami átszámítva körülbelül 5000 forint, vagyis egy teljes irodai csomagért legalábbis versenyképes ajánlatnak mondható.

Az oktatási intézmények számára a StarOffice ingyenes.

A StarOffice magyar nyelvű változatával nem egyhamar fogunk találkozni, ám Linux és Windows alá egyaránt elkészült az OpenOffice 1.1-es kiadásának honosított változata. A csomagot bárki szabadon letöltheti az FSF.hu alapítvány oldaláról.

➲ http://www.openoffice.hu
➲ http://www.staroffice.com

e-kormányzati keretrendszer

Az Open Source Software Institute (OSSI) elkészült Project Leopard néven futó fejlesztésének első változatával. Az e-kormányzati webszolgáltatás-keretrendszer alapját a Linux, az Apache, a MySQL és a PHP/Perl/Python adja – a kezdőbetűk alapján LAMP-nak rövidítik. Feladata keretrendszerjellegéből fakadóan nem az, hogy tényleges alkalmazásokat bocsásson az érdekeltek rendelkezésére, hanem hogy egységes, az együttműködést elősegítő, gyors alkalmazásfejlesztést lehetővé tévő hátteret biztosítson a hivataloknak, akik önállóan építhetik fel az igényeikhez igazodó számítástechnikai rendszert.

A OSSI ezzel egy időben létrehozta a Open Government Interoperability Standard (OGIS) munkacsoportot is, amely a kormányzat, a vállalati és az oktatási szektor képviselőinek részvételével a Leopard keretrendszer segítségével fejlesztett alkalmazásokra és modulokra vonatkozó irányelveket fogja kidolgozni.

Az elektronikus kormányzati rendszerek létrehozása világszerte napirenden van, ugyanakkor számtalan probléma megoldását is igényli. A Leopard tervezet révén egyelőre az alapok fektethetők le, megteremthető az a háttérrendszer, amely biztosítja az adatbázisok egységességét és az alkalmazások közötti együttműködés lehetőségét. A célok között természetesen a költségek leszorítása is szerepel, amelynek elérésében a nyílt megoldások alkalmazása csak az egyik tényező; a könnyű felügyelhetőség, a gyors bevezetés és a zökkenőmentes együttműködés lehetősége egyaránt segíti a hivatalok olcsóbb fenntartását.

➲ http://www.oss-institute.org
➲ http://leopard.sourceforge.net

A támogatásban van az üzlet?

Az IBM Global Services egy éven belül világszerte elérhető támogatást kíván nyújtani a linuxos asztali gépekhez, derült ki az első Linux Desktop Consortium konferencián. Mivel a felhasználók egyre inkább érdeklődnek a Linux használata iránt, a jelenleginél kiterjedtebb, általánosabb célú műszaki támogatási szolgáltatásra van, illetve lenne szükség. Az IBM nem egyedül dédelget ilyen terveket, a SuSE Linux 210 millió dolláros áron való felvásárlását bejelentő Novell ugyancsak átfogó, folyamatosan elérhető támogatási szolgáltatást kíván nyújtani vállalati ügyfeleinek. Ugyanakkor a háttérben sokféle összefonódást figyelhetünk meg, hiszen az IBM ötvenmillió dollárt kíván a Novellbe befektetni – abba a cégbe, amely ma már elsősorban szolgáltatásokból és tanácsadásból él. Valószínűsíthető, hogy a piacot valamilyen formában felosztó két nagy cég szerepvállalásának köszönhetően a Linux-terjesztések hozzáférhetősége a magánszemélyek számára egy kicsit romlik, ugyanakkor a vállalati igényeket a felgyorsuló fejlesztések és a bővülő szolgáltatások révén magasabb szinten tudják majd kielégíteni.

Felmérések szerint a Linux részesedése az asztali gépek operációs rendszereinek piacán 2006-ra a jelenlegi 1,5 százalékról 7 százalékra nő. Az IBM és a Novell hiánypótló tevékenységükkel erősen javíthatják a Linux elfogadottságát.

Robotépítés nyíltan

Egyszerre meglepő és megmosolyogtató az Open Automaton Project (OAP) célkitűzése: nyílt robotépítési eszközgyűjtemény létrehozása, amelynek alapján amatőrök is értelmes, önállóan működő robotokat építhetnek. Az alkotók szeretnék megszüntetni azt a szakadékot, amely az amatőrök által barkácsolt robotok és a jobban felkészült kutatók által összeállított gépek minősége és tudása között észlelhető. Az érdeklődőket kapcsolási rajzokkal, leírásokkal és forráskódokkal egyaránt segíteni szeretnék. A költségek féken tartása érdekében – a kitűzött költséghatár 1500–2000 dollár – a gép összeállításához bárki számára hozzáférhető, szabványos elemeket, leginkább PC-s környezetbe illeszkedő alkatrészeket ajánlanak.

➲ http://oap.sourceforge.net

Elbűvölő noteszek

Az Apple új, PowerPC G4 processzorokkal felszerelt iBook-modelleket mutatott be.
A 12"-os és 14"-os kijelzővel készülő gépekbe 256 MB DDRRAM, CD-íróként és DVD-olvasóként egyaránt használható optikai meghajtó és 32 MB memóriával ellátott ATI Mobility Radeon 9200 VGA-vezérlő kerül, operációs rendszerük pedig a Mac OS X 10.3-as, Panther kódnevű változata. Mindegyik gép AirPort Extreme csatolóval rendelkezik, amely a 802.11g szabvány szerint működik, vagyis 55 Mb/s sebességű kapcsolatot kínál. Kiegészítőként Bluetooth-csatolót is kérhetünk, amely például az Apple új, vezeték nélküli billentyűzetének vagy egerének csatlakoztatására használható. Az új modellek súlya kiépítéstől függően 2,5–3 kg, akkumulátoruk pedig 6 órán keresztül képes árammal ellátni őket. Az alapváltozat ára átszámítva kb. 250 000 forint, ezt a bővítésekkel természetesen 350 000 forint fölé is könnyedén feltornászhatjuk.
➲ http://www.apple.com

Fizess vagy viszlát!

Red Hat Enterprise Linux és Fedora Project: tessék választani! Ezt javasolja mindenkinek a Red Hat, amely 2004. április 30-án beszünteti a Red Hat Linux jelenlegi formában történő támogatását. A vállalati változatot azoknak a felhasználóknak ajánlják, akik megbízható, biztonságos, támogatással ellátott Linux-változatot szeretnének futtatni gépükön – és ezért hajlandóak is fizetni. A Red Hat legolcsóbb ajánlatához 179 dolláros ár tartozik, ezért viszonylag ritkán megjelenő, ám várhatóan alaposan tesztelt, a szükséges frissítésekkel öt évig javítgatott terjesztést lehet majd kapni. A Fedora Project teljesen nyílt, ingyenes, fejlesztését a közösség végzi, és értelemszerűen mindenki úgy szerez hozzá támogatást és frissítéseket, ahogy tud.
A Red Hat lépése érzékenyen érintheti azokat, akik korábban megszerették a kalapos terjesztést, ugyanakkor a viszonylag gyakran megjelenő terjesztések összeállítását és hosszabb időre szóló támogatását térítés nélkül vállalni egyenlő az üzleti csőddel. Ha a Red Hat fel akarja venni a versenyt a Microsoft termékeivel és támogatásával, akkor beláthatjuk, hogy pénzt kell kérnie a saját szolgáltatásaiért. A Microsoft az őskövületnek számító Windows NT 4.0 rendszereket még jelenleg is támogatja, ezzel a vállalati felhasználóknak kiszámíthatóságot és egyfajta biztonságot ad. A ritkábban megjelenő vállalati Red Hat-terjesztéseket kevesebb erőforrás bevetésével is lehet majd támogatni, valamint az öt évre ígért szolgáltatás hasonló tervezhetőséget biztosít a vállalati ügyfeleknek. Az ingyenélők pedig senkinek nem kellenek…
➲ http://www.redhat.com

Zenét tessék!

A legendássá – és a Roxio általi felvásárlása nyomán üzleti vállalkozássá – vált Napster már feltöltőkártyákat is

kínál a fiataloknak. A 15 dollárért megvásárolható, akár ajándékként is adható kártyák 15 zeneszám letöltésére jogosítják fel tulajdonosukat. A megoldás legfőbb előnye – ne feledjük, fiatalokról van szó – az, hogy a letöltéshez nem kell hitelkártyával rendelkezni. A Napster 2.0 szolgáltatás nemrég indult, egy-egy zeneszám letöltéséért 99 centet kell fizetni, egy album ára vagy a havi előfizetés díja pedig 9,95 dollár.
A Napster üzemeltetőinek nem kisebb üzleti ellenféllel kell szembenézniük, mint az Apple, amely iTunes szolgáltatását immár a Windows operációs rendszert futtató felhasználóknak is elérhetővé tette. A Macintosh-tulajok számára már korábban is hozzáférhetővé vált szolgáltatás indulásakor példátlan sikert aratott, ami a windowsos táborban is megismétlődött: az ügyfélprogram első egymillió letöltésére alig három és fél napot kellett várni. Az Apple a Pepsivel is szövetkezett, az amerikai üdítőitalfogyasztók a kupakokban hamarosan zeneszámok letöltésére jogosító kódokat találhatnak. A Pepsi százmillió szám ilyen módon való szétosztását tervezi. Ugyancsak ide kapcsolódik, hogy a Wal-Mart is zeneletöltő szolgáltatás beindítását tervezi. A kiskereskedelmi óriáscég körülbelül kétszázezer zeneszámot fog hamarosan elérhetővé tenni az interneten, tervei szerint a versenytársai által kért 99 centes ár alá vágva. A nagyjából 220 forintos, kedvezőnek mondható dalonkénti ár tehát a verseny beindulásával remélhetően még csökkenni is fog.
➲ http://www.napster.com

Xandros Desktop 2.0

A Xandros bejelentette a Xandros Desktop 2.0 kiadását. A terjesztés alapját a Debian Linux 4.0 adja, efölé a 3.1.4-es KDE került; a meglévő és esetleg nem

linuxos környezetbe való beépülést pedig a Microsoft alapú hálózatok támogatása és a szinte nélkülözhetetlennek mondható OpenOffice 1.1-es változata segíti. A Xandros igazi asztali rendszernek szánja termékét, így már a telepítést is igyekszik gyerekjátékká egyszerűsíteni, illetve minden általános feladat elvégzését lehetővé teszi a géppel, így beépített CD-író programot, csevegési lehetőséget és multimédiás alkalmazásokat kínál. A Deluxe Edition vásárlói egyben a CrossOver Office 2.1-es kiadásának egy xandrosos változatához is hozzájutnak, vagyis végszükség esetén Microsoft Office-t, Macromedia Dreamweavert, Adobe Photoshopot és további windowsos alkalmazásokat is futtathatnak gépükön. A normál kiadás ára 39 dollár, ehhez 30 napos elektronikus levélben elérhető támogatás jár.
A Deluxe Editionért 89 dollárt kell fizetni, ehhez 60 napos támogatást nyújtanak. Aki már megvásárolta valamelyik korábbi kiadást, az igényeinek megfelelően több frissítési lehetőség közül is választhat. Az ismerkedést a 30 napos próbaváltozat teszi lehetővé, amely a cég weblapjáról tölthető le.
➲ http://www.xandros.com

Beceneve: hatékonyság

Iparági szereplők hosszú sora biztosította támogatásáról a Transmetát és új, Efficeon névre keresztelt processzorát. Az alacsony fogyasztású Crusoe lapkájával nagy figyelmet, ám mérsékelt piaci sikert elért cég saját bevallása szerint tanult a hibáiból, és lapkájának fej-

lesztésekor ezúttal nemcsak a kis áramfelvételre, de a versenyképes teljesítményre is kellő hangsúlyt fektetett. A legfőbb ellenfél természetesen az Intel, amely Centrino processzoraival teljesítmény és alacsony fogyasztás terén is – némi képzavarral élve – magasra tette a mércét. Azt mindenesetre már tudni, hogy az előzetes próbák alapján azonos fogyasztás – de nem azonos órajel – mellett az Efficeon gyorsabb, mint a Centrino. Az Efficeon – elődjéhez hasonlóan – egy köztes kódformáló réteggel alakítja a 32 bites processzorokra írt programokat a saját nyelvére. Belsőleg a Transmeta processzorai 256 bites szóhosszal dolgoznak, és a kódformálás révén egyszerre nyolc 32 bites utasítást képesek végrehajtani. A kódformáló réteg egyben optimalizálást is végez – éppen ezért rendkívül fontos szerepet játszik. A Transmeta mérnökei az új sorozattal módosították a réteg működését, amely immár nem a legjobb kód kialakítására törekszik, inkább ésszerű, gazdaságos programfuttatási megoldásokat keres. Az Efficeon mindössze 79 millió tranzisztort tartalmaz, így várhatóan gyártásánál is nagy kihozatalt lehet majd elérni. Ha ez teljesül, akkor jó eséllyel akár 100 dolláros áron is megvásárolhatjuk az új processzort, amely hamarosan a gyártók kínálatában is megjelenik.
➲ http://www.transmeta.com

Tévételefon

A Vodafone hamarosan megkezdi a Sharp V601SH típusú mobiltelefonjának árusítását a Távol-Keleten – a készüléket nem kevesebb, mint 2 millió képpont felbontású fényképezőgéppel látták el. Igaz, hogy hasonló képességekkel bíró telefont már lehet a Föld túlsó felén venni, ám a Sharp terméke kisebb, kamerája autofókuszos, megfelelő kábellel pedig az általa rögzített fényképek és mozgóképek lejátszása céljából tévékészülékre is csatlakoztatható. A Vodafone hamarosan újabb újdonsággal kívánja elkápráztatni ügyfeleit, V601N jelzésű telefonja a telefonáláson túl a hagyományos tévéadások vételére is alkalmas lesz. A 24 mm vastag és 119 gramm súlyú telefon a tévéadások legfeljebb 30 másodperces részleteinek rögzítésére is képes, a mozgóképek tárolását MPEG-4 formátumban végzi, valamint állóképek kilopását is lehetővé teszi. A mindennapi nehézségek elkerülése érdekében hagyományos tévéantennával is használható, illetve a bemenetet videómagnóról és videókameráról is képes fogadni.

Vérdíj a vírusírók fejére

A Microsoft ötmillió dolláros alapot hozott létre a számítógépes vírusokat vagy férgeket készítő személyek kézre kerítésének ösztönzésére. Jutalmat az kaphat, aki valamilyen kártékony program készítőjének elfogását és elítélését érdemben segítő információkat szolgáltat. A kezdeményezést az FBI és az Interpol is támogatja, mivel a vírusok problémája az internetes terjesztés miatt az egész világot érinti. Jelenleg két felhívás van érvényben, a Blaster és a Sobig nevű férgek készítőit keresik, a jutalom mindkét esetben 250 000 dollár. Azt, hogy kizárólag a Windowsra írt vírusok készítőinek a skalpjáért lehet-e jutalmat kapni, nem említik.

Vízjel feketében

A Hitachi kutatói újfajta elektronikus vízjelezési megoldást dolgoztak ki fekete-fehér képekhez. A színes képek vízjellel való ellátása régóta lehetséges, ám a fekete-fehér nyomatok hasonló védelemmel való ellátása sokkal nehezebb feladatot jelent – ugyanakkor lemondani sem lehet róla, hiszen mindennapi életünk során sokkal több fekete-fehér nyomattal találkozunk, mint színessel. A színes képek rengeteg adatot, árnyalatokat, általában kifinomult formákat tartalmaznak, így egyrészt van mire építeni a vízjelet, másrészt az elrejtése sem okoz gondot. A csupán feketével készült betűk és egyszerű ábrák ellenben viszonylag kevés adattal leírhatók, és a vízjelnek az eredeti tartalom eltorzítása nélkül való elhelyezése is gondot jelent. A Hitachi mérnökei ezért megvizsgálták, hogy a befogadó személyek hogyan olvassák és értelmezik a szövegeket és egyszerű ábrákat, például a grafikonokat; mely területek azok, amelyeket úgy lehet módosítani, kismértékben torzítani, és így a vízjel elrejtésére használni, hogy az olvasók számára a védelem észrevétlen maradjon. Eljárásukat például üzleti dokumentumok, szerződések hitelességének megerősítésére lehet majd használni, de nyilvántartási adatok beágyazására is alkalmazható lesz. Az új megoldás a felhasználók számára várhatóan a Hitachi Keymate/Mark alkalmazásának következő változatában válik elérhetővé.

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

A SCSI rétegen való áthaladásnak köszönhetően Jeff számos olyan szolgáltatásra támaszkodhatott, amelyet egyébként saját magának kellett volna kódolnia.

diff-u

Bevezettek egy új rendszerhívást: a `tgkill()` eljárást, amely az olyan homályos hibák kezelésére szolgál, amikor egy folyamatnak küldött jelzés egy teljesen más folyamatnál köt ki. *Molnár Ingo* alkotta meg a hívást, a tgkill nevet pedig *Linus Torvalds* javasolta – a név a hívás bemeneti adataira utal, azaz a szálra és a csoportra (t: thread, g: group). A régi hívás (`pthread_kill()`) nem akadályozta meg a jel eltévelyedését okozó programhibát, és a bemeneti adatai csak szálazonosítók voltak.

Eugene Weiss megírta a Submount programot, amely a merevlemezek gyorscseréjét támogatja. A programnak része a `subfs` modul, amely egy ál-fájlrendszert hoz létre a kívánt beillesztési ponton. A modul ezután a fájlrendszerműveletek előtt és után elvégzi a szükséges befűzési és leválasztási műveleteket. Így az alkatrészt az adatsérülés veszélye nélkül bármikor el lehet távolítani.

Elkészült az OpenPosix tesztkészlet 1.0.0-s változata. Ez a csomag a jelek, üzenetsorok, jelzők, időzítők és folyamatütemezők alapvető Posix-megfelelőségének a vizsgálatára szolgál. Igaz, nem éppen a Linuxra jellemző eszköz, és maga a Linux nem helyez olyan nagy hangsúlyt a Posix-megfelelőségre, mint más operációs rendszerek, de azért az OpenPosix tesztkészlet nagyon hasznosnak bizonyulhat azokon a területeken, ahol a Linux ügyel az együttműködésre.

Martin Schlemmer rájött, hogy az OSS működik az ICH5-ös jelelosztón (Intel I/O Controller Hub), ha az ICH5-ös azonosítókat felveszi a rendszermag forrásában. Ez valóban segíthet egyes rendszereken, de amint *Jeff Garzik* is rámutatott, nem minden ICH5-ös esetén működik.

A QLogic teljesen újraírta a Fibre Channel (FC) vezérlőprogramot a saját ISP21xx/ISP22xx/ISP23xx lapkái és HBA-i számára. Az új vezérlőprogram egyáltalán nem támogatja a 2.4-es rendszermagot, ellenben jelentős teljesítményjavulás érhető el vele. A QLogic célja az, hogy a programot bejuttassák a hivatalos 2.5-ös fába a 2.6-os fejlesztésének megkezdése előtt.

A Serial ATA (SATA) előkészítése során Jeff Garzik írt egy vezérlőprogramot, amellyel az ATA-lemezek elérhetővé válnak SCSI-felületen keresztül. Jeff szerint a SCSI sajátosságai és az új rendszermag-szolgáltatások, például a SysFS iránti kiemelt programtámogatás miatt ez a program remek SATA-támogatási alap lesz. A SCSI rétegen való áthaladásnak köszönhetően Jeff számos olyan szolgáltatásra támaszkodhatott, amelyet egyébként saját magának kellett volna kódolnia.

Molnár Ingo bejelentette, hogy elkészült az Exec Shield adatbiztonsági szolgáltatás, ami védelmet nyújt számos (bár nem az összes) lehetséges hibaforrás ellen. A védelem kiterjed a vermek, az átmeneti tárak és a függvénymutatók túlcsordulására, illetve sok egyéb sebezhető pontra. Ráadásul mindehhez nincs szükség a felhasználói alkalmazások újrafordítására. Habár az Exec Shield nem teljes megoldás, más biztonsági intézkedésekkel együttesen használva nagyon hatékonynak ígérkezik.

Zack Brown

Linux Journal 2003. október, 114. szám

Linux-index

1. A Python-kérések száma 2003 májusában: **11,9 millió**
2. Ennyi ezer különböző kiszolgáló küldött Python-kéréseket 2003 májusában: **325**
3. A japán kormány ennyi millió dollárt (188 millió jent) fog költeni az új, Linux alapú IBM/Oki bérszámfejtő rendszerre: **1 590 000**
4. Ennyi japán kormányalkalmazott számára végez majd bérszámfejtést az új rendszer: **800 000**
5. Ennyi milliárd dollárt (350 milliárd jent) tesz ki Japán jelenlegi bérszámfejtő rendszerének éves költsége: **2,96**
6. Várhatóan ennyi milliárd dollár megtakarítást jelent a japán kormány számára az új, Linux alapú rendszer bevezetése: **1,48**
7. Ennyi dollárt költ évente a dél-afrikai kormány a jogdíjas programok engedélyeztetésére: **352**
8. Ennyi éve használ Linuxot a kanadai nemzeti vasúttársaság (Canadian National Railway): **10**
9. Ennyi milliárd dollár származik majd a vezeték nélküli LAN-berendezések eladásából 2006-ra: **4**
10. Ennyi ezer szakember rendelkezik a Linux Professional Institute tanúsítványával: **27**
11. Ennyi ezer a (Linux alapú) TiVo-ügyfelek száma: **750**
12. Várhatóan ennyi millió háztartásban lesz majd a TiVóhoz hasonló PVR (személyi videófelvevő) 2006-ban: **30**
13. A hirdetők ennyi százaléka tervezi a televíziós reklámkeret csökkentését a PVR várható elterjedése miatt: **76**
14. Ennyi ezer Linux alapú HP-munkaállomást használ a DreamWorks a filmek előállításához: **1**
15. Ennyi ezer Linux alapú processzort alkalmaznak a DreamWorks leképezőtelepei: **3**
16. Ennyi millió sor kódot ültetett át Linuxra a Pixar: **300**
17. A szélessáv-használat elterjedésének aránya Koreában: **70%**
18. A szélessáv-használat elterjedésének aránya az Egyesült Államokban: **35,9%**
19. Az Apache részesedése a vezető webkiszolgálók piacán idén júliusban: **63,72%**

Források
1–2.: *Guido van Rossum*
3–6.: Reuters
7.: InfoWorld
8–10.: A CanadianBusiness.com a Forrester Research jelentésére hivatkozva
11–13.: A BusinessWeek a Forrester Research jelentésére hivatkozva
14–16.: eWeek
17–18.: WebSiteOptimization.com
19.: Netcraft (➲ http://www.netcraft.com)

Linux Journal 2003. október, 114. szám

Rendszermag-fejlesztési hírek

Linus Torvalds újabb lépést tett a következő megbízható változat felé. A 2.5-ös sorozat lezárult, és a végső 2.6.0-s változat előkészítéseként elindult a 2.6.0-s próbaváltozat. A kisebbik változatszám növekedésével egyidejűleg Linus azt reméli, hogy majd szigorúbban bírálhatja el a beérkező javítókészleteket. Habár nincs szó a kód befagyasztásáról, Linus leszögezte, hogy a nagy változtatásokat valószínűleg visszautasítja majd, kivéve bizonyos különleges eseteket. Szándékai szerint a 2.6.0-s változat még az év vége előtt megjelenik.
A Linux történetében a leghosszabb és legkeservesebb vitákat a lezárás okozza. A felhasználók éveken át könyörögtek azért, hogy pusztán a lefordított rendszermagból ki lehessen nyerni valahogyan a fordításhoz használt beállításokat. Több javítókészletet is javasoltak, de végül is úgy tűnik, hogy a 2.6-os változat fogja ezt megvalósítani. *Randy Dunlap* kódjával a felhasználók felfedhetik a beállításokat egy */proc* felületen keresztül, magát a konfigurációs fájlt hozzácsatolhatják a rendszermag bináris állományához, illetve ezek helyett akár egyszerűen eldobhatják az adatokat. Randy javítókészlete egy darabig bizonytalanul lebegett az *Alan Cox*-féle fában, mígnem 2003. július végén bevették Linus fájába.
Most már egyszerűen megvalósítható a titkosított fájlrendszerek beillesztése hurokeszközön keresztül, annak köszönhetően, hogy *Andries Brouwer* és mások átdolgozták a `cryptoloop` programot. A `BLK_DEV_CRYPTOLOOP` beállítás engedélyezésével átlátható módon beilleszthetők a titkosított fájlrendszerek. Egy fájlcsoport titkosítása olyan egyszerűvé válik, mint egy könyvtárfa átmásolása egyik helyről a másikra. Mivel mindez hurokeszközön keresztül történik, az is könnyen megoldható, hogy a rendszernek csupán egy kis részét titkosítjuk, a kevésbé kényes anyagok pedig nyitottak maradnak. Egyedül az jelenthet gondot, hogy – a 2003. augusztusi állapotok szerint – a `cryptoloop` megváltoztathatja a hurok API-t, ami miatt forrásszintű változtatásokra van szükség minden olyan illesztőprogram esetében, amely a hurokeszközt használja.
Új rendszermag-illesztési program jelent meg 2003 júniusában a Synaptics TouchPad eszköz számára, amely nagyrészt a kapcsolódó XFree86 illesztőprogramon alapul. A *Peter Osterlund* nevéhez fűződő illesztőprogram egy háromgombos, két görgetőgombbal ellátott egeret emulál: támogatja a többujjas használatot, a függőleges és vízszintes irányú görgetést, a szélgörgetést vonszolási műveletek alkalmával, a tenyérérzékelést és a sarokérintést. Pillanatnyilag minden jel arra utal, hogy a 2.6.0-s változat kiadása előtt bekerül a hivatalos fába.
Daniel Stekloff megírta és kiadta a *libsysfs* könyvtárat, amelynek segítségével kényelmesen kezelhetővé válnak a SysFS-felületek. Elege lett abból, hogy kétszer kellett megírnia ugyanazt a kódot minden egyes, a SysFS kezelésére felkészített alkalmazásban. A libsysfs nagymértékben leegyszerűsíti a SysFS fájlrendszerhez való csatlakozást. A DevFS-t felváltó `udev` – *Greg Kroah-Hartman* kódja – egyike a legjelentősebb fejlesztéseknek, amelyek jelenleg a *libsysfs*-t használják, de úgy gondolom, hamarosan sokkal több ilyen lesz. Daniel is részt vett az `udev` tervezésében a fejlesztés korai szakaszában – ez szolgált Greg munkájának alapjául.
David Howells megalkotta a takaros CacheFS fájlrendszert, amelyben minden blokkszervezésű eszköz lemeznek látszik, s így bármely más fájlrendszer is használni tudja őket. Noha eredetileg az AFS fájlrendszer támogatására szánták, David úgy tervezte meg a CacheFS-t, hogy független legyen a fölötte működő fájlrendszertől. A CacheFS minden bizonnyal hamarosan bekerül a rendszermagba, hiszen Linus Torvalds már régóta szeretett volna valami ehhez hasonlót.
Úgy tűnik, hogy a CacheFS kiválóan alkalmas fájlrendszer alapú változatfelügyeleti szolgáltatások ellátására, amely pillanatnyilag nagyon foglalkoztatja Linust. Mindazonáltal a kód elég későn jelent meg a nem megbízható változatban, ezért további felhasználói visszajelzésekre lesz szükség ahhoz, hogy a CacheFS-t bevegyék az alaprendszerbe.

Zack Brown

Linux Journal 2003. november, 115. szám

David úgy tervezte meg a CacheFS-t, hogy független legyen a fölötte működő fájlrendszertől.

LME konferencia

November 8-án szombaton került megrendezésre a Linux-felhasználók Magyarországi Egyesületének V. GNU/Linux Szakmai konferenciája.
Az egyesület történetében immár második alkalommal sikerült kiemelkedően elegáns helyszínen megrendezni az év legfontosabb nyílt forráshoz kapcsolódó szakmai találkozóját. *Sári Gábor*, az egyesület elnöke délelőtt 10 órakor köszöntötte a több mint ötszáz látogatót a budapesti Novotel Szállóban. Ezt követően egyszerre öt helyszínen kezdődtek meg az előadások.
Idén szinte minden olyan cég képviselte magát támogatóként, előadóként, aki valamilyen formában kötődik a Linuxhoz.
Mint minden hasonló rendezvényen, természetesen itt is megjelent az LME CD-író projektje, ahol a nap folyamán több száz linuxos korong talált gazdára.
A rendezvény utolsó mozzanataként húsz értékes ajándék került kisorsolásra a látogatók között. A rendezvény végén, az általános jó hangulat közepette végzett gyors felmérés szerint a támogatók, a kiállítók, valamint a látogatók egyöntetűen az eddigi legsikeresebb LME-rendezvénynek találták a V. GNU/Linux Szakmai konferenciát.

FSF.Kolozsvár

Olvasóinknak már nem kell bemutatnom az FSF.hu Alapítványt, hiszen rendszeresen jelenik meg róluk hír, hogy ilyen vagy olyan kezdeményezést támogatnak, fordításokat szerveznek, bemutatókat tartanak. Októberben egy felkérésnek tettek eleget, amikor a határon túlra, Kolozsvárra látogattak el, egy egynapos rendezvényre, ahol a szabad programokról tartottak előadásokat. Ezt a rendezvényt immár az ország számos pontján megtartották (április óta hét alkalommal), de ez volt az első eset, hogy a csapat a határon túlra merészkedett. Nagy örömömre velük tarthattam. Talán elsőre elgondolkodtató, hogy miért is van szükség arra, hogy az embereknek bemutassuk a szabad programok előnyeit és hátrányait. Igen, úgy gondolom, hogy mindkét oldalt be kell mutatnunk, hiszen nem az a lényeg, hogy „eladjuk” ezeket a termékeket, hanem az, hogy megmutassuk, mi vár az emberre, ha a szabad oldalon tapogatózik. A másik nagyon fontos szempont, hogy megértsük a szabad világ alaptételeit: ismerjük és értsük meg egymást. A Linux nem egy dobozolt kereskedelmi termék, amit az ember megvesz, majd ha nem működik, visszaviszi a boltba, és addig csapkodja az asztalt, amíg vissza nem kapja a pénzét. Itt az ember ugyanolyan halandókkal veheti fel a kapcsolatot, kérhet tőlük tanácsokat, fejlődhet velük együtt, mint önmaga. Ezért is hívják sokan „szabad közösségnek” e programok használóit. A legtöbbet pedig akkor tesszük e közösségért, ha mi is jól érezzük magunkat benne: programokat, túrákat, bulikat szervezünk – ebben pedig az FSF.hu élen jár.

Ez a bemutató sem zajlott másként, az összesen négy napra széthúzott útba (amiből kettőt kocsiban töltöttünk) belefért másfél nap szabadidő is. És ha már Kolozsvár, akkor Torda és Torockó meglátogatása kötelező program. Tudom, hogy néhány képpel lehetetlen visszaadni a táj gyönyörűségét, de azért kedvcsinálónak talán elég. Emellett még Kolozsvár is megér egy külön misét, hiszen a még meglévő épületremekek, a templomok és nem utolsósorban a Mátyás-szobor egyszerűen mesébe illő. A román zászló színeire festett játszóterek és utcai szemetesek is csak egy kicsit tudnak rontani az összhatáson.

De térjünk vissza az előadásokra. Az előadók érintették a biztonság, a használhatóság, az együttműködés kérdéseit, kitérve mind az asztali, mind a kiszolgálóoldali kérdésekre. Igaz, röviden, de az idő csupán ennyit engedett. Érdekes volt, amikor a „melyik rendszert válasszam” kérdés vetődött fel. Bár az előadók igyekeztek függetlenként megszólalni, nem rejtették véka alá a saját kedvenceik nevét.

A legtöbb kérdés itt is két témakör körül forgott: egyrészt a szakmai támogatás, másrészt a programok beszerzése körül. Hogy miként lehet szakmai támogatást kialakítani, mindig is kemény dió, hiszen a jó szakemberek kinevelése bizony sok időt és munkát vesz igénybe. Ami pedig az elérést illeti: a nálunk már egyre jobb internetellátottság Kolozsvár felé még lényegesen hézagosabb. Sajnos ez egy saját farkába harapó kígyó, hiszen a legtöbb segítséget az ember vagy egy közeli barátjától kapja, vagy a neten keresztül. Reméljük, szép lassan a határ túlsó oldalán is „fejlődik” mind a két terület.

Sajnos, amíg a helyi közösség el nem jut egy megbízható szintre, addig az üzleti élet természetszerűleg el van zárva a szabad programok elől: bármennyire is gazdaságos, hatékony, költségkímélő, ügyes, szép, jó, ha nincs egy szakember, akit vészhelyzet esetén biztosan ki tud hívni a cég, akkor egyetlen ügyvezető sem igen vállal be egy új rendszert. Ugyanez volt megfigyelhető Budapesten is pár évvel ezelőtt, sőt amíg az említett ügyvezető nem találkozik személyesen legalább két-három szakértővel, általában nem is érdeklődik igazán a szabad világ iránt.

És természetesen az előadások után szigorúan kötelező jelleggel csapatépítés is zajlott, ahol kötetlenebb körülmények között ismerkedhettünk egymással. A csapatépítés az egyik helyi pizzéria különtermében folyt, a már említett okok miatt pedig igyekeztünk jól érezni magunkat.

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

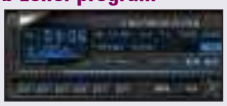
Közönségdíjak 2003

Köszönet mindenkinek, aki részt vett a szavazáson – íme, az eredmények!

Az idei Linux Journal Közönségdíj szavazása során – többnyire az eszközök terén – néhány új kategóriával találkozhattunk, néhány régebbit pedig töröltünk – hála a böngészők beépített, előugró ablakok (pop-up) megjelenését gátló szolgáltatásának. A négyhetes, elektronikusan elérhető idei szavazáson a tavalyinál többen vettek részt: csaknem 7500-an. Az eredmények vegyesebbek a „Kedvenc változat" kategóriában, de a tavalyi év nyerteseinek nagy része idén is visszaköszönt.

A legkedveltebb zenei program

1. xmms
2. noatun
3. mpg123



Az xmms továbbra is uralja e kategóriát, az idén zsinórban harmadik éve foglalja el a vezető helyet. A hivatalos listán töltött első évében a noatun a második helyet vívta ki. A szavazók által megadott új kedvenc az MPlayer.

A leggyakrabban használt biztonsági segédprogram (Backup Utility)

1. tar
2. Amanda
3. Arkeia

A kedvenc munkaállomás kategória mellett a kedvenc biztonsági segédprogram tűnik a legkevésbé piacfüggőnek. A tavalyi nyertesek ismét visszatértek: a `tar`, az Amanda és az Arkeia végeztek az élen. Az `rsync` a szavazók által beküldött új favorit. Szerencsére közületek csak kevesen gondolták úgy, hogy a biztonsági másolatok nyápicoknak valók.

A legnélkülözhetetlenebb Linux-könyv

1. *Ellen Siever és mások*:
   A Linux dióhéjban (harmadik kiadás)
2. *Vicki Stanfield és Roderick W. Smith*
   Linux-rendszerfelügyelet
3. *Matt Welsh és mások*:
   A Linux futtatása (harmadik kiadás)

Az első három helyezett ismét a régi, bár a Linux-rendszerfelügyelet és A Linux futtatása idén helyet cserélt. Sok „puritán" felhasználó csupán a súgóoldalakra (man pages) támaszkodik. A Linux Journal irodájába érkező levelek és az új kiadások bírálatainak a számából ítélve a linuxos könyvpiac felfutóban van.

A legjobb processzor

1. AMD Athlon
2. Intel Pentiums
3. AMD Opteron

Az idei évben robbant be a mezőnybe a közönségdíjra pályázó Opteron: egyből a harmadik helyet érte el kategóriájában. A tavalyi harmadik helyezett, az AMD Duron az ötödik helyre esett vissza, a PowerPC lett a negyedik. Harmadik éve zsinórban az Athlon a kedvenc.

A kedvenc internetböngésző

1. Mozilla
2. Konqueror
3. Galeon

Itt is megegyeztek a győztesek tavalyi év nyerteseivel, de a Konqueror és a Galeon 2003-ban helyet cserélt. A Netscape népszerűsége egyre csökken, a 6.x a 7362-ből már csupán 236 szavazatot nyert el ebben a kategóriában. A Firebird több figyelmet kapott a szavazók által javasolt sorrendben, az összesített listán a hetedik helyet érte el.

A kedvenc Linux Journal-rovat

1. Cooking with Linux (a Linuxvilágban a Fogadó a Linuxhoz néven olvasható)
2. Kernel Korner (Szaktekintély)
3. Paranoid Penguin (Szaktekintély)

Ah, Marcel; il est un homme savant, gentil and trés drôle. Már miért is ne tisztelnénk azt, aki a rendszergazdák okítása mellett a borok világába is elkalauzol bennünket?

A legnépszerűbb adatbázis

1. MySQL
2. PostgreSQL
3. Oracle 9i

Tavaly megtorpant egy kicsit az Oracle, amikor az InterBase lett a harmadik helyezett, de az idén visszatért az első három közé. Bár a MySQL megtartotta vezető helyét, a PostgreSQL egyre jobban felzárkózik. A Firebird, az InterBase-kódra alapozott független adatbázis, ismét a kedvenc a szavazók által javasolt sorban.

Kedvenc asztali munkaállomás

1. Homemade
2. Monarch AMD 2000+ System Special
3. Los Alamos ULBx

Lehet, hogy ezt az új kategóriát „nyílt forrású tákolmányversenynek" kellett volna nevezni. Bár a Monarch, a Los Alamos, az Apple G5, a Dell és néhány Sun masina is kapott szavazatokat, a voksolók csaknem kilencven százaléka ebben az új kategóriában a Homemade-et nevezte kedvenc munkaállomásának. Képzeljétek csak el, mit lehetne mindebből kihozni: két négyfős csapat, száz dollár, két nap és egy szemetesládányi használt alkatrész!

A kedvenc fájlcserélő rendszer

1. Gnutella
2. Freenet
3. MORPHEUS

Valahányszor azt látom, hogy Gnutella, a Nutella jut eszembe, az a fincsi európai mogyorós kakaóvaj. Tény, hogy a Gnutella is szinte ugyanilyen jó. Az idén a Freenet betört az első három helyezett közé, visszalökve ezzel az audiogalaxyt az ötödik helyre, míg az első és a harmadik helyezett megegyezik a tavalyival. Az eDonkeyval, a Kazaával, az mldonkeyval és a Bit Torrenttel viszont tele van a szavazók javaslatlistája.

A kedvenc Linux-változat

1. Debian
2. Red Hat
3. Mandrake Linux

Most először vívta ki magának az első helyet a Debian ebben a kategóriában – egy olyan évben, amikor a csapból is a Linux asztali felhasználása és szupergyors, és könnyű telepítési kellékei folytak. A technofilek lázadásáról van szó, vagy az `apt-get` csábítja az új felhasználókat?

A programozók itala

1. Kávé
2. Víz
3. Tea

E kategória szépsége a szavazók által beírt kedvencek listája, ahol az egyéni ízlés pompás és érdekes nyomait leljük fel. Az első három hely mindig a kávéé, a vízé és a teáé, de a szavazók által beírt italok a vodkától a pálinkáig, a Yerba Mate-től a Tangig terjedtek. Az idei év legvisszataszítóbb itala a kávé fahéjjal és pirospaprikával. Utóirat: gratulálunk annak a szavazónak, aki helyesen betűzte a Merlot-t, r-rel és t-vel!

A leggyakoribb levelezőügyfél

1. Evolution
2. KMail
3. Mozilla

Idén is sikertelen maradt a GUI levelezőügyfelének az első három helyezett ellen intézett támadása. A negyedik helyezett muttnak a Mozilla szavazatainak a felét sikerült megszereznie. Az Evolution megjelenésének második évében mindösszesen 151 szavazattal múlta fölül a KMailt.

A legismertebb beágyazott Linux

1. Qtopia
2. MontaVista
3. Lineo

2003 már a második esztendő, hogy a „beágyazott” kategóriában is lehetett szavazni – az idén kétszer annyi szavazat született, mint tavaly. A Qtopia és a MontaVista helyet cserélt, és a beágyazott SnapGear piacon való megjelenése évében egyből a negyedik helyre ugrott. A szavazók által beírt legnépszerűbb a tenyérgépekre (PDA) írt Familiar projekt.

A legnépszerűbb grafikai program

1. Gimp
2. ImageMagick
3. CorelDraw

Ismét a Gimp nyert – igen, igen, mi is meg vagyunk lepve. Ami ennél is érdekesebb, azoknak az Adobe Photoshopot megadóknak a száma, akik azt a CrossOver Office-on keresztül használják.

A kedvenc linuxos játék

1. Frozen Bubble
2. Quake 3
3. TuxRacer

Szórakoztatóbb, mint egy *Michael Jackson*-botrány és veszélyesebb, mint egy ügyvéd, aki a szellemi tulajdont próbálja meghatározni: a Frozen Bubble egyből az első helyre ugrott a hivatalos listán. Vajon leváltja-e jövőre a szavazók egyes számú idei jelöltje, a Neverwinter Nights?

A kedvenc munkakörnyezet (Desktop Environment)

1. KDE
2. Gnome
3. Window Maker

A kedvenc változat kivételével ez a kategória kapta a legtöbb szavazatot, ami érthető, hiszen a Linuxot az asztali gépekhez reklámozták leginkább. A szavazatok 44 százaléka a KDE-re esett, amely immáron hatodik éve az első helyezett. A Gnome 23 százalékkal tartja a második helyet. Az Ion a szavazók új kedvence, a tavalyi favorit fluxbox a negyedik lett a hivatalos listán töltött első évében. És milyen jó, hogy valaki beírta megjegyzésként: „Egyik sem, mind gyogyi!” – már kezdtük azt hinni, hogy idén ez elmarad.

A kedvenc üzenőügyfél (Instant-Messaging Client)

1. Gaim
2. Jabber
3. Kopete

Három az egyhez ismét a Gaim nyert. A második és harmadik helyet a tavalyi lista esélyesei töltik be. A tavalyi év befutója, a Licq csupán 388 szavazattal az ötödik helyre esett vissza. Idén az irssi a szavazók listán kívüli jelöltje. Bár nem sokat tudok a fegyverekről, abban egészen biztos vagyok, hogy egy tizenkétlövetű Mossberg 500A nem üzenőügyfél.

A legkedveltebb programozási nyelv

1. C++
2. C
3. PHP

Nosza, mindenki csapjon gyorsan a billentyűk közé! Pár másodperc, és indul a vita! A tavalyi év nyerteseinek és befutóinak fordított listáját látjuk: a C++ mindössze 23 szavazatos fölénnyel 2003-ban az első helyre került. A Perl kiszorult az első három helyezett közül, ez a közönségdíj történetében először fordult elő. A C-vonalon a C# a szavazók kedvence.

A kedvenc irodai programcsomag

1. OpenOffice.org
2. AbiWord
3. KOffice

2003-ban az irodai csomag (Office Suite) és a szövegszerkesztő (Word Processor) csoportot ebbe az általánosabb kategóriába vontuk össze. Ezt a szavazást nagy előnnyel

az OpenOffice.org nyerte meg. A 6650 szavazat közül 4317-et kapott az OpenOffice.org, 477 szavazattal követi az AbiWord. Az ingyenes, extrákkal teli irodai programok meglehetősen jók. Továbbá hadd gratuláljunk a szavazóknak az igencsak kisszámú MS Office-javaslatért – mindenki átállt már vagy csak nem mertek beszélni róla?

A leggyakrabban használt fejlesztőeszközök

1. GCC
2. KDevelop
3. Emacs

A Gimphez hasonlóan a GCC is minden évben jelentős előnnyel nyer a kategóriájában – majdhogynem megsajnálja az ember a versenytársakat. Bár az Emacs ismét a harmadik helyet kapta, a tavalyi második helyezett Kylix a hatodik helyre csúszott vissza, a KDevelop pedig visszatért az első három közé. Az idén a szavazók kedvencjelöltje az Anjuta, amely a Glade-et a szövegszerkesztő kellékekkel és egy egyedi IDE-szimulátorral ötvözi.

A kedvenc hálózati és kiszolgáló eszközök

1. Cyclades AlterPath ACS
2. CommuniGate Pro
3. SnapGear SME 550

A piac kínálta tömérdek termék hívta életre ezt a kategóriát 2003-ban. A szavazók jelöltlistája azt mutatja, hogy számos termék illik ebbe a csoportba. Úgy tűnik, sokan használnak Cyclades AlterPath ACS-t (advanced consoler server) a hálózatok távoli karbantartásához; s a Linksys útválasztók is számos felhasználó szívéhez állnak közel.

A legkedveltebb szerkesztő

1. Vim
2. vi és klónjai
3. GNU Emacs

Tavaly a Vim kétszer annyi szavazatot kapott, mint a vi – az idén ez a különbség háromszoros. A harmadik helyezett biztosan tartja magát idén, mivel az Emacsot kedvelők egyre több mindenre tudják használni a programot. De mi lett Elvis iránti érzelmeinkből? Röpke 14 szavazatra futja? Kate viszont a hivatalos listán töltött első esztendejében a negyedik helyig eljutott.

A legnépszerűbb kiszolgáló

1. SGI Altix 3000
2. IBM DB2 OLAP Server
3. Tyan Thunder K8S

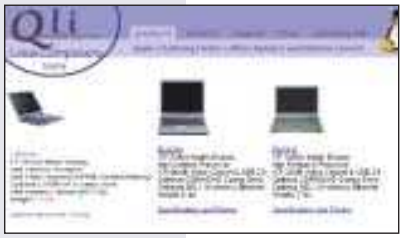
Ezt az idei februári király Altix címlapsztorinknak köszönhetik, ugye? Vagy a szavazók a tavaly januári LinuxWorld New York 2003 rendezvényen botlottak bele az ott kiállított, nagy Altix-gépekbe? Bárhogy is legyen, egy biztos: mindenki imádja az Altix 3000-et. A Dell, az HP, a Sun és a Compaq található még a szavazók jelöltlistáján, míg néhányan azon méláznak, hogy miért venne az ember kiszolgálót, amikor akár építhet is egyet? – a házi gyártású kiszolgálók viszik el a pálmát a szavazók jelöltlistáján.

A rendszergazdák nélkülözhetetlen rendszerkarbantartó eszközei

1. Webmin
2. YaST
3. Ximian Red Carpet

Mivel a „sysadmin" kellékek széles kínálata beszerezhető, gondoltuk, itt az ideje, hogy megkapják a saját kategóriájukat. Az önműködő frissítőcsomagok nyerték el a legtöbb szavazatot. A szavazatok több mint harminc százalékát a Perl alapú Webmin kapta, és ezzel nyert is. Távoli másodikként kullog mögötte a SuSE YaST-ja. Természetesen sok szavazó azt írta, hogy nekik nincs másra szükségük, mint egy parancssorra és a vi vagy a vim valamiféle kombinációjára ahhoz, hogy ellássák a rendszerfelügyeleti feladatokat.

A legnépszerűbb hordozható munkaállomások



1. QLi 15" AMD notebookok
2. QLi Pentium 4 notebookok
3. QLi Centrino notebookok

Ismét egy új kategória az idei díjazottak között: a legnépszerűbb hordozható munkaállomásé, amely távolról sem kapott annyi szavazatot, mint más csoportok. A szavazatok többsége a szabadon jelölhető sorból érkezett. Mindenki egy laptopot választhatott kedvencéül, bár a Zaurus számos szavazatot kapott. A szavazók jelöltlistáján a Dell és az IBM laptopjai és az Apple PowerBookja is szerepel.

A legnépszerűbb linuxos weboldal

1. Slashdot
2. LinuxFR
3. Freshmeat

Az elmúlt néhány évben ebben a kategóriában volt a legszorosabb a verseny. A Slashdot mindössze 343 szavazattal ütötte ki ki a LinuxFR-t az első helyről – ez lenyűgöző, különösen, ha azt vesszük, hogy 6588 szavazónk volt. Jó volt látni, hogy az idén nem kergült meg mindenki és nem ajnározta a „Da Linux French Page"-et. A Slashdot áll tehát az élen, ha valaki szupergépekről, kaliforniai kormányzójelöltekről vagy kétórás Cadillac-árreklámokról – akarom mondani: Mátrix-folytatásokról – kíván olvasni.

Linux Journal 2003. november, 115. szám

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

A hónap szakmai tanácsai

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

A szabálytalan leállítás összekuszálja a nagyméretű könyvtárakat

Létezik egy könyvtárunk, amely nagyjából százezer kis- és közepes méretű fájlt tartalmaz. A Red Hat 8 telepítése után gondjaink támadtak egy nem szabályos rendszerleállás következtében. Egy „túl nagy fájlleíró" hibaüzenetet kaptunk, majd a rendszer egyfelhasználós üzemmódba lépett, és csak az `e2fsk` program párbeszédes üzemmódjában lehetett kijavítani a hibát. A könyvtár teljesen eltűnt, és az összes állomány a *lost+found* könyvtárba került. Soha nem volt ilyen gond a Red Hattel a 7.3-ig bezárólag. Két különböző számítógépen is meg tudtuk ismételni a jelenséget, mindkettő Red Hat 8-at futtatott. Akár `ext2`, akár `ext3` fájlrendszert használunk, a hiba fennáll.
Joe Waytula, joseph.waytula@ipaper.com

Valószínűleg az történt, hogy a rendszer akkor állt le váratlanul, amikor éppen abba a könyvtárba írt valami. Emiatt a könyvtár ellentmondásos (inconsistent) állapotba került, és elveszett. Bár az `ext3` naplózó fájlrendszer, ez csak azt elenti, hogy a rendszer gyorsan magához tér, nem azt, hogy nem léphet fel adatvesztés. Ebben az esetben a könyvtár volt az elvesztett adat. Az `ext2` és `ext3` a hasított (hashed) könyvtárkiterjesztés nélkül nem kedveli a sok állományt tartalmazó könyvtárakat. Lassan kezeli őket, és emiatt egy összeomlás következtében hosszú időszakban felléphet az adatvesztés esélye. Javaslom, hogy térjetek át a ReiserFS, XFS vagy JFS használatára, ezek ugyanis jobban támogatják a nagy könyvtárakat.
Marc Merlin, marc_bts@google.com

termcap vagy terminfo?

Szeretnék `xterm`-en keresztül egy távoli SCO-géphez kapcsolódni. Nem találom azokat a beállítási adatokat, amelyekkel a SCO terminálja `xterm` alatt utánozható lenne. Az `xterm terminfo`-t vagy `termcap`-ot használ?
Aldo Gentile, agentile@lacapital.com.ar

Minden korszerű Linux-terjesztés `ncurses`-t használ, amely a `terminfo`-t részesíti előnyben a `termcap`-pel szemben. Próbáld ki az `rxvt` programot a `vt100` vagy a `vt200` terminéltípus beállításával.
Jim Dennis, jimd@starshine.org

Indításkor a Dell noteszgép lefagy

Vettem egy Dell D800 noteszgépet 256 MB memóriával és 30 GB merevlemezzel. A hajlékonylemez-meghajtó és a CD-meghajtó felváltva használható benne. Windows XP volt előre rátelepítve, de akárcsak az asztali gépemen, a lemez egy részén Red Hat 8.0-t szerettem volna futtatni. A Windowsnak 6 GB helyet hagytam, a maradék területet a Linux szükségletei szerint osztottam fel. A Grub rendszerindítóval értem el, hogy a Linux és a Windows felváltva indulhasson el. Úgy tűnt, hogy mindent meg tudok oldani, amit az asztali gépen is sikerült. A telepítés simán ment a CD-ről. Nem készítettem indítólemezt, mert nem akartam a CD-meghajtót kivenni. Gratuláltam magamnak a sikeres telepítéshez, de az újraindítás után az a kellemetlenség ért, hogy a gép indítás közben lefagyott. A billentyűzet meghalt, és semmi sem adott életjelet. A képernyő nem sötétült el, de ki kellett kapcsolnom a gépet, és a Windowst kellett elindítanom. Kétszer is megismételtem a telepítést, de mindig ugyanaz lett az eredmény. Mintha egy megszakítás állítaná meg a gépet. Olvastam már az APM-megszakítások és a noteszgépek kapcsolatáról, de nem tudom, hogy most mi a teendő. Linuxot akarok a gépemre!
Rob Borochoff, borochoff456@comcast.net

Ha bármikor kérdés vetődik fel a Linux és egy adott noteszgép együttműködésével kapcsolatban, az első dolgunk legyen megtekinteni a kitűnő Linux on Laptops webhelyet: ➲ http://www.linux-laptop.net.
Greg Kroah-Hartman, greg@kroah.com

Töltsd le vagy vedd meg a Knoppix-terjesztést (➲ http://www.knoppix.com), amelyet nem kell telepíteni, mert a CD-ről fut. Nézd meg, hogy elindul-e, és milyen alkatrészeket ismer fel magától. Ha ez működik, akkor értékes adatokhoz juthatsz a Red Hat vagy más terjesztés beállításához.Azért hirdetem a Knoppixot, mert jelen pillanatban ennek a legjobb az alkatrész-felismerő és -beállító képessége.
Jim Dennis, jimd@starshine.org

Ethernetkártyák elnevezésének lehetőségei

Két ethernetkártyát használok a gépemben. Lehetséges-e, hogy az egyik mindig az `eth0`, a másik mindig az `eth1` nevet kapja?
Ning Qian, nq6@columbia.edu

Természetesen lehetséges. Erre való a `nameif` program, amely a hálózati eszközöket a MAC-címük alapján ismeri fel és nevezi el. Ezért teljesen mindegy, hogy a rendszermag milyen sorrendben kérdezi le a kártyákat, mindig ugyanúgy tudod elnevezni őket. Ha a `nameif` programot az eszköz csatlakoztatásakor elinduló parancsfájlból (lásd ➲ http://linux-hotplug.sf.net) indítod, akkor a hálózati eszközök megfelelő módon lesznek elnevezve, akármikor találja is meg őket a rendszermag.
Greg Kroah-Hartman, greg@kroah.com

Létezik az `ether=` rendszermag-parancssori kapcsoló, ezzel a rendszermagba fordított illesztőprogramok által vezérelt etherneteszközöknek lehet nevet adni, illetve más értékeket átadni. A betölthető modulként fordított eszközmeghajtókat parancssori segédprogramokkal és a */etc/modules.conf* állományon keresztül lehet módosítani.
Jim Dennis, jimd@starshine.org

Ha különböző kártyákról van szó, használj modulokat, és a sorrendet a */etc/modules.conf* állományban határozd meg:

```
alias eth0 e100
alias eth1 3c59x
```

Marc Merlin, marc_bts@google.com

Új termékek

Pioneer DVD-felvevő

A Pioneer Electronics moduláris, Linux alapú DVD-felvevőt dobott piacra, amelyet az ipari videófelhasználóknak szánnak. A PRV-LX1 valós időben készít videófelvételeket, segítségével olyan egyszerűen

végezhető el a DVD-anyag felvétele, tömörítése, szerkesztése és rögzítése, mint egy hagyományos videómagnóval. Az eszköz több forrásból gyűjti össze a hang- és képjeleket a DVD-re, és lehetőséget ad a DVD-menük és a fejezetek belépési pontjainak szerkesztésére. Az alaprendszer egy DVD-R/RW-meghajtót és 120 GB-os merevlemezt tartalmaz, de egy második DVD-R/RW is megrendelhető hozzá. A videokimenetek és -bemenetek: videokomponens, kompozit, S-video és DV; a hang pedig: kétcsatornás kiegyenlített, kétcsatornás nem kiegyenlített és koaxiális digitális. A PRV-LX1 ezen kívül a következő csatlakozókkal bír: RS-422A, ethernet, VGA-kimenet, USB2, külső szinkronjel és fejhallgató.
➲ http://www.pioneerelectronics.com

Columbitech Wireless Suite

A Columbitech bejelentette a Columbitech Wireless Suite-hoz való linuxos ügyfél megjelenését.
A Columbitech Wireless Suite lehetővé teszi a tenyérgép-alkalmazásfejlesztők, a gépjárműgyártók és mások számára, hogy olyan vezeték nélküli biztonsági alkalmazásokat hozzanak létre, amelyek nem igényelnek nagy teljesítményű processzort vagy sok memóriát. A csomag három részből áll. A Columbitech Wireless VPN a noteszgépre vagy a tenyérgépre telepíthető ügyfélprogram; a Columbitech Gatekeeper a kiszolgálóra telepített DMZ, amely a hitelesítést és a terheléselosztást végzi; végül a Columbitech Enterprise Server, amit a vállalati hálózaton egy kiszolgálóra szükséges telepíteni, és a VPN-munkameneteket kezeli. A Columbitech Enterprise Server tartalmazza a Columbitech VPN Servert és a Columbitech WAP Connectort, képes VPN kapcsolattartásra és WTLS alapú WAP-kommunikációnak.
➲ http://www.columbitech.com

Lone-Tar 4.0

A Lone Star Software Corporation bejelentette a Lone-Tar biztonsági mentést és helyreállítást végző programjának 4.0-s változatát. A 4.0-s újdonságai: 256 bites titkosítás; biztonsági mentés optikai adathordozóra, például DVD-RAM-, DVD-R/RW- és CD-R/RW-lemezekre; HP szalagos meghajtók ismerete; önműködő eszközfelismerés; licenckezelő; áttervezett grafikus és karakteres felhasználói felület; online frissítés; és önkicsomagoló telepítővarázsló. Ezenkívül a Lone-Tar 4.0-val rendszerindításra képes Lone-Tar mentőlemezt lehet készíteni a Rescue-Ranger vészhelyzet utáni helyreállítással, azaz egy lemezen egyszerre lehet a mentést és a helyreállítást végezni. Az eszközkezelő segítségével a felhasználók bármikor eszközöket adhatnak hozzá, cserélhetnek ki vagy távolíthatnak el, és bármelyik tömörítési és titkosítási beállításait megváltoztathatják.
➲ http://www.cactus.com

iNAV 9200

Az iNAV 9200 többprotokollos átjáró több kommunikációs protokoll közötti közvetítésre képes önálló eszköz.
Ez az 1U magas rendszer arra lett tervezve, hogy olyan hálózatokat kössön össze, amelyek nem beszélnek közös nyelvet, ilyenek például a régi típusú hurokkapcsolt hálózatok és az újabb csomagalapú hálózatok. Az iNAV 9200 kimondottan széles sávú (DSL, kábel, vezeték nélküli) elérést megkívánó alkalmazásokhoz lett tervezve. Használható egyedülálló, többszolgáltatásos kapcsolóként vagy médiaátjáróként, illetve képes elvégezni a csomagok útválasztását és osztályozását, ATM-kapcsolást, ATM-szétdarabolást és ATM-összerakást is tud. Az OEM-ek választhatnak több hurokkapcsolt csatolófelület közül, például T1/E1/J1, T3/E3, OC-3/STM-1 és OC-12/STM-4.
➲ http://www.iphase.com

NemeSys grafikus munkaállomások

A RackSaver bemutatott egy felső kategóriás grafikus munkaállomáscsaládot, amelyet a tervezés, tartalomkészítés, megjelenítés és más nagyszámú grafikai műveletet kívánó alkalmazáshoz fejlesztettek ki.
A NemeSys 720-as munkaállomássorozat két Xeon vagy Opteron processzort és legfeljebb 16 GB memóriát tartalmaz. Egy munkaállomásba legfeljebb nyolc merevlemez szerelhető, így a tárkapacitás 2 terabájt is lehet. A NemeSys munkaállomások támogatják a felsőkategóriás grafikus kártyákat, például az nVidia Quadro FX kártyacsaládot, beleértve az új nVidia Quadro FX 3000-es kártyát is. Sokféle alaplap állítható be a NemeSyshez, például Tyan, Intel, Arima, MSI és SuperMicro. DVD-R, DVD-ROM, CD-R/RW és CD-ROM beépítése lehetséges.
➲ http://www.racksaver.com

Lindows.com BusinessStation

A Lindows.com új ajánlata a BusinessStation, egy kis karbantartás-

igényű számítógép, amelyet munkaterminálként, nyilvános hozzáférési pontként, kereskedelmi adatpultként lehet használni ott, ahol fontos a webalapú kapcsolattartás a vevőkkel, alkalmazottakkal, illetve a háttérrendszerekkel. A BusinessStation a Lindows.com WebStation gépén alapul, és szerepel benne egy olyan hálózati beállítóeszköz, amellyel 1–5000 gépet lehet bármilyen webböngészőből beállítani. A BusinessStation lelke egy LindowsCD, amelyen webböngésző, üzenetküldő, médialejátszó, webalapú levelező és irodai csomag található.
➲ http://www.lindows.com

Linux Journal 2003, 115. szám

A SARS vírus genetikai állományának megfejtése

2003. április 7-én 1 órakor érkezett meg a SARS vírus tenyészete a Michael Smith Genomtudományi Központba. Öt nappal később a laboratórium elsőként hozta nyilvánosságra a vírus genetikai állományának nukleotidsorrendjét.

Idén áprilisban a Genomtudományi Központban (Genome Sciences Centre, azaz GSC) tettük közzé az első teljes génkészletszerkezetét annak a coronavírusnak, amely ismereteink szerint a Severe Acute Respiratory Syndrome (SARS) járvány okozója. A GSC-nél az 1999-es kezdetek óta minden vizsgálatot, tárolást és hálózati hátteret Linux-rendszerek alatt végeznek. A SARS vírus projektben az adatok tárolását, feldolgozását és nyilvánosságra hozását számos Linux-kiszolgáló végezte, kezdve a pehelysúlyú, de hasznos IBM x330-tól egészen a behemót nyolcutas Xeon x440-ig. A Linux által nyújtott rugalmas háttér lehetővé tette, hogy a megfejtési folyamat szinte minden lépését automatizáljuk. A Linux-közösség támogatásával és a hírcsoportok, webes cikkek és HOGYAN-ok segítségével hihetetlenül olcsón munkára tudtuk fogni a középkategóriás alkatrészeket.

A SARS első dokumentált megjelenése óta (2002. november 16.) a vírust összesen 8458 esetben észlelték Kínában (92%), Kanadában (3%), Szingapúrban (2%) és az Észak-Amerikai Egyesült Államokban (1%), valamint több mint 25 egyéb országban. A SARS halálozási esélye közelítőleg 5–10%, a 60 felettiek esetében azonban 50% körüli. 2003. június 24-re a SARS már 807 életet követelt, igen mély negatív hatást gyakorolva az érintett régiók gazdaságára – egyedül Kína több milliárd dollárt veszített turisztikai és adójövedelmeiből.

2003. március 27-én ***Marco Marra***, központunk igazgatója és ***Caroline Astell***, projektünk vezetője úgy döntött, hogy megfejti a SARS coronavírus genomjának szekvenciáját. 2003. április 7-én éjjel 1 órakor egy torontói páciensből származó vírus, a Tor2 izolátum genetikai anyagának közel 50 ng-ja érkezett a kanadai Winnipeg 4. szintű Nemzeti Mikrobiológiai Laboratóriumából. Öt nappal később, 2003. április 12-én a Tor2 (Tor2/SARS) coronavírus genetikai állományának 29751 nukleotidhosszúságú része már felkerült Apache kiszolgálónk Zope/Plone alapú oldalára – elérhetővé téve azt a teljes nyilvánosság számára. Néhány nappal később az úgynevezett Urbani izolátum szekvenciáját küldte el a CDC (Centers for Disease Control) Atlantából, Georgia államból.

A biológia virágzásnak indul

Az 1990-es évek előtt nem létezett olyan módszer, amellyel nagy mennyiségű nukleotidsorrend-adatot gyorsan meg lehetett volna határozni. Az Emberi Genom Projekt (Human Genome Project, azaz HGP) 1991-ben kezdődött, és 1999-re a sorrendnek mindössze 15 százalékát sikerült megfejteni. Ugyanakkor az 1990-es években kifejlesztett új módszereknek hála a HGP gyorsan közeledett a befejezés felé. 2000 közepe táján az emberi szekvencia kilencven százaléka már elérhető volt, és mostanra az emberi génállomány nukleotidsorrendje lényegében rendelkezésünkre áll. A HGP-hez hasonló projektek eredményei nyilvánosan is elérhetők az NCBI GenBank oldalain.

Működésének első tíz éve során (1982–1992) a Génbank valamivel több mint 100 MB-nyi szekvenciát gyűjtött össze 80 ezer bejegyzésben. A következő évtizedben (1992–2002) a GenBank rakétasebességgel növekedésnek indult, és az adatbázis elérte a 29 GB-ot – az emberi genom tízszeresét – 22 millió bejegyzésbe szedve. A Génbank minden nap tízezer bázissorrendadatot kap a világ különféle laborjaitól. Az egyik ilyen labor a GSC, amely 2003. április 13-án jelentette be a GenBankban a Tor2/SARS szekvenciáját. Ha kíváncsiak vagyunk, hogy milyen szerepet játszott a Linux abban folyamatban, amely végül a GI:29826276 számú bejegyzés megszületéséhez vezetett, vissza kell nyúlnunk a kezdetekig.

Parancssoros bioinformatika

Valósítsunk meg egy kis bioinformatikát a bash és néhány másik, a */bin* és */usr/bin* könyvtárban bujkáló program segítségével. A Tor2/SARS genom GC arányát fogjuk kiszámítani – azaz a G-C vagy C-G bázispárok részarányát. Hogy érdekes legyen a dolog, az `awk` programot nem fogjuk használni. Először is `wget`-tel töltsük le a sorozatot, a `-q` kapcsolóval csillapítva a kimenetét:

```
> wget -q
➥http://mkweb.bcgsc.ca/sars/AY274119.fa
> head AY274119.fa
gi|30248028|gb|AY274119.3| SARS coronavirus
➥TOR2
ATATTAGGTTTTTACCTACCCAGGA...
```

A nukleotidsorrend-fájlok FASTA formátumban vannak, amely a fejlécsort és magát a rögzített hosszúságú sorokra osztott nukleotidláncot tartalmazza. A következő kód megszámolja, hogy hány G és C található a láncban, majd az eredményt az összes bázis arányában jeleníti meg:

```
> grep -v "^>" AY274119.fa | fold -w 1 |
tr "ATGC" "..xx" | sort | uniq -c |
sed ´s/[^0-9]//g´ | t -s "\012" " " |
sed ´s/\([0-9]*\) \([0-9]*\)/scale = 3;
➥\2 \/ (\1+\2)/´ |
bc -i
scale = 3; 12127 / (17624+12127)
.407
```

Szekvenciánk 29 751 bázisából tehát 12 127 elem lesz akár G vagy C, így a GC-tartalom 41%-ra adódik.

1. kép Szekvenálólaborunk panorámája: *1.* folyamatoknak megfelelő vonalkódok, *2.* a Tango folyadékkezelő felület, *3.* -80° C-os mélyhűtők, *4.* áramforrások a PCR (polimeráz láncreakció) készülékekhez, *5.* ABI 3730XL szekvenátorok, *6.* ABI 3700 szekvenátorok, *7.* x330 vezérlőfürt, *8.* hálózati és áramcsatlakozók *9.* a szekvenátorok ventilátorjáratai

0-18 TB három év alatt

1999 júniusában a labor hat szép bézsszínű számítógépet és közel ugyanannyi embert alkalmazott. A központi fájlkiszolgáló (2×Pentium 3, 400 MHz, 512 MB RAM, Red Hat 5.2 és 2.0.36-os rendszermag) három RAID-0 18 GB SCSI-merevlemezt kezelt DPT IV kártyán keresztül. Újabb 50 GB programozott RAID került a második gépbe (Pentium III, 400 MHz). További három Linux-ügyféllel és egy Microsoft Windows NT állomással együtt alkották a BC Cancer Agency (BCCA) hálózatát. Megszületésünk időpontja nagy előnyünkre szolgált. Mint minden kutatólabor, mi is lemezeket osztunk meg, folyamatokat kezelünk, programokat fordítunk, valamint adatokat tárolunk és kezelünk. Más szavakkal: éppen olyasmit csinálunk, amiben a Unix kiváló. Ha két-három évvel korábban kezdünk, az akkor még ifjú Linux bevezetése nem lett volna könnyű. Így ma valószínűleg ahelyett, hogy az olcsó kiöregedett PC-inket irodába száműzzük vagy egyéb kevésbé nagyfokú hálózati feladatokra osztjuk be, megpróbálhatnánk a legjobb árat kapni az igen jelentős összegekbe került, már kiöregedett Sun kiszolgálóinkért. Szerencsére kiderült, hogy lehetőségünk van viszonylag olcsó PC-ket vásárolni, majd Linuxot telepítve rájuk nagyméretű, rugalmas és elképesztően költséghatékony Unix-környezethez jutnunk. A Linuxnak köszönhetően többé már nem volt szükség rá, hogy egy ember fizetését Unix-munkaállomásokra költsük.

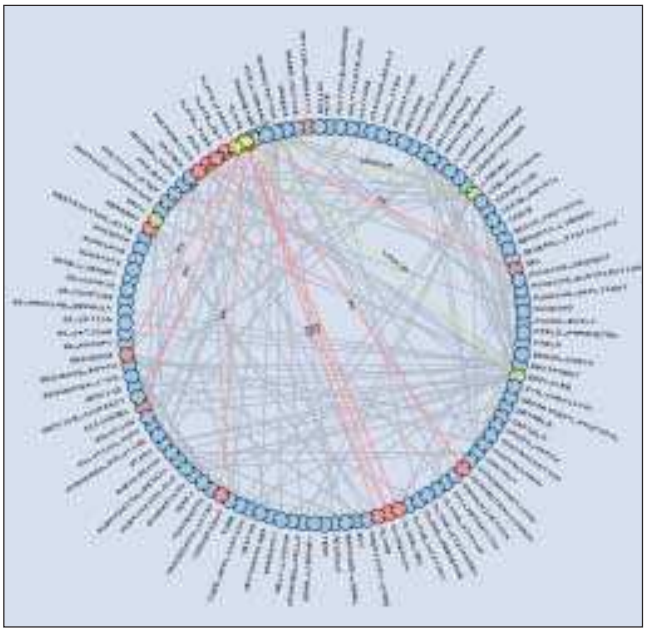
Éppen jó időben választottuk a Linuxot. A 2.0-s rendszermag sziklaszilárd volt; az NFS kiszolgáló megerősödött, és teljes értékű asztali környezetek között válogathattunk. A létfontosságú bioinformatikai analízis-eszközkészleteket letölthettük és lefordíthattuk. Ilyen például a nyílt forrású HGP: BLAST (sorrend-összehasonlító), a Phred (a szekvenátor által készített jelek értelmezése), a Phrap (sorrendek összeállítása) és a Consed (nukleotid- és aminosavsorrendek összeállításainak megjelenítése), továbbá néhány nukleinsav és fehérje-adatbázis. Természetesen a Perl volt a „mindenes” ezekben a műveletekben. A számítástechnikai munka elindításához igen kevés pénzt használtunk fel, így a nagy összegeket sokkal hatékonyabban költhettük a labor fejlesztésére (*1. kép*).

A Linux elcsípi a SARS-t

1999 őszén megkaptuk első automata DNS-szekvenátorunkat, egy MegaBACE 1000-est (*6. kép*). A szekvenátor segítségével egy DNS-mintában megállapítható a nukleotidok sorrendje, a módszer azonban jelenleg 5–800 bázis meghatározására korlátozódik egy időben. Ez az olvasási hossz jóval kisebb a jelenleg ismert legkisebb genomnál is (a Tor2/SARS mérete harmincezer nukleotid). Ezért az automata szekvenátor egyszerre 96 mintát kezel; vannak olyan típusok is, amelyekben egyszerre több, 96 vagy 384 mintahelyet (vályút) tartalmazó speciális lemez is elhelyezhető.

2. kép Első nemzedékbeli kiszolgáló-alkatrészek: *1.* VA Linux VAR900 2xXeon-500 1 TB felkínált tárhellyel, *2.* Raidion.u2w RAID-vezérlők, *3.* 2x8x36 GB SCSI lemez és *4.* VA Linux 2230-asok és 3x10x72 GB SCSI lemez

A MegaBACE egy SCSI-eszköz, az Applied Biosystems (ABI) 3700 és 3730XL szekvenátorok pedig (*6. kép*) soros felületen keresztül kezelhetőek, az adatokat viszont ethernetkapcsolaton keresztül küldik. Nagy mennyiségű adatot gyűjtenek önműködően, a hozzájuk tartozó program viszont egy mutass és kattints (point-and-click) Windows-alkalmazás. Az ABI gépek a hozzájuk adott helyi Oracle adatbázisba mentik az adatokat. Egy Unix alapú vezérlőprogram forradalmasítaná e gépek kihasználását, különösen a nagyobb laboratóriumokban. Máris sikerült csökkentenünk a 3700-es karbantartási munkáit azáltal, hogy az eredetileg a szekvenátorral szállított PC-t IBM x330-as gépre cseréltük (6. kép). A Windows alapú szekvenálórendszernek a linuxos hálózatunkba történő beillesztése remek munka volt az `smbmount`, az `rsync`, a Perl és az Apache számára. Az operátor minden egyes sorrend-meghatározási kör befejezésekor beindítja a webvezérlésű adattükrözési folya-



3. kép LIMS sémánk bemutatása. A lemeztábla (sárga) négy táblára hivatkozik (zöld), rá pedig 14 tábla hivatkozik (vörös)

4. kép A vonalkódokat hálózatba kötött Zebra nyomtatók készítik (bal oldalt). A LIMS hordozható felületét iPAQ-ek biztosítják (jobb oldalt)

matot, és az új adatokat a hálózati lemezekre másolja.
Tükrözés után az állományokat először a nyers szekvenálójel-formátumból átalakítjuk a tényleges nukleinsavbázisok jelévé és a hozzájuk tartozó minőségértékekké (a meghatározás biztonságának mértéke), majd MySQL-adatbázisban (3.23.55max) tároljuk őket. Ezzel a módszerrel eddig kétmillió sorrendet rögzítettünk, azaz körülbelül 1 TB nyers nukleotidsorrend-adatot.
A MySQL Laboratory Information Management System (LIMS) adatbázis központi szerepet tölt be a nukleotidsorrend megállapításának folyamataiban. Sémájában 115 táblát, 1171 mezőt és 195 idegen kulcsot találunk. Az adatbázis az összes, a laborral kapcsolatos összetevőt, felszerelést, folyamatot és műveletet követi. Különleges alkalmazáslogika és elnevezési szabályok segítségével sikerült áthidalnunk a MySQL hiányosságát, miszerint nem rendelkezik beépített idegenkulcs-kezeléssel. Az idegen kulcsokat `FKTYPE_TABLE__FIELD`-nek nevezzük, jelezve, hogy egy `TABLE_FIELD`.-re mutatnak a `TABLE` táblában. Az idegen kulcs nevének elhagyható `TYPE` részét akkor használjuk, ha több kulcs

5. kép A laborban szinte minden vonalkódokkal van ellátva

is mutat ugyanarra `TABLE_FIELD` mezőre.
A labor szakemberei vonalkódolvasóval kiegészített Wi-Fi Compaq iPAQ gépekkel tartják a kapcsolatot a LIMS adatbázissal (*4. kép*). Az iPAQ-ok a belső, saját `mod_perl` készlettel bővített Apache webkiszolgálónkra csatlakoznak. A különféle objektumok, azaz a megoldások, lemezek és felszerelések vonalkóddal vannak ellátva (*5. kép*). A vonalkódokat hálózatba kötött Zebra S600/96XiIII vonalkódnyomtatóval készítjük nagy ragadóképességű címkékre (4. kép), amelyek -80 °C (-112 °F) hőmérsékletű hűtőinkben is fennmaradnak. A vonalkódkészítő program Perlben íródott, a címkék formázására a ZPL nyomtatónyelvet használja, a nyomtatást pedig `lpr`-en keresztül osztja meg.
A MegaBACE 1000-es óta laboratóriumunkban a szekvenátorok három nemzedéke fordult már elő, és jelenleg már három ABI 3700-es és három ABI 3730XL gépet (6. kép) üzemeltetünk. A legfrissebb, az ABI 3730XL több 384 mintahelyes lapot képes befogadni, és 1152 DNS minta nukleotidsorrendjét határozza meg 24 óra alatt. Minden egyes minta 700–800, nagy biztonsággal azonosított bázist jelent. Egyetlen 3730XL körülbelül 800 ezer bázist olvas le naponta.
A Tor2/SARS genom nukleotidsorrendjének a megállapítását az úgynevezett teljes genomra irányuló (whole-genome shotgun, WGS) módszerrel végeztük. Ennél a megközelítésnél a genom véletlenszerűen kiemelt szakaszait szekvenáljuk redundáns módon, majd utólag állítjuk össze a teljes genomsorrendjét. Tekintve, hogy a szóban forgó vírus méretét körülbelül 30 ezer bázisra becsültük, a teljes genom leolvasásához legalább negyven szekvenciameghatározást kellett végrehajtani. Minthogy azonban a leolvasás véletlen régiókból történt, a minimális olvasásszámnál többet kellett végrehajtanunk, hogy elég átfedésünk legyen a teljes összeállításhoz. A redundancia miatt biztosabbak is lehetünk benne, hogy a genom egyes pozícióin valóban az adott bázist tartalmazó nukleotid áll.

6. kép
Szekvenátorok: 1. MegaBACE 1000, 2. a szekvenátor PC-je, 3. szünetmentes áramforrás, UPS, 4. a szekvenátor áramforrása, 5. ABI 3700-es szekvenátorok, 6. ABI 3730XL és 7. x330 fürt

A bézs besötétedik

Amikor első IBM x330 kiszolgálóinkat vásároltuk, amelyek ma már egy 168 CPU-t tartalmazó fürt részei (*7. kép*), az 1U felület volt a kereskedelmi off-the-shelf (COTS) kategória határa, ahonnan kezdve élvezni lehetett a COTS árait. Bézs-színű gépeinket többé már nem használjuk megosztott számításokra. A komoly terhelésnek alávetett termelési rendszereink, azaz az Apache és a MySQL, az IBM 4U x440s-eiben kaptak helyet, ezekben a nyolcutas hiperszálakkal (hyperthreading) és 8 GB memóriával ellátott Xeon-csomópontokban. A gépeken SuSE 8.1 fut – ez azon kevés terjesztés egyike, amelyik képes kezelni az IBM Summit lapkakészletét. A x440-es NUMA típusú gép, ahol négyprocesszoros modulonként 32 MB L4 gyorstár található, így az IBM Summit foltja nélkül a rendszermag csak két CPU-t lát. A SuSE 2.4.19 rendszermagja bigmem+Summit támogatással mind a nyolc processzor és a 8 GB memória használatát lehetővé tette. Ezek az x440-esek még a 2.5-ös rendszermagsorozatban megjelenő fejlett NUMA ütemező nélkül is igen hasznos igavonónak bizonyultak, és lehetővé tették, hogy nyolc BLAST folyamatot futtassunk párhuzamosan, miközben elegendő memóriánk marad a teljes emberi genom gyorstárazására a megosztott memóriában. Bárki, aki azt állítja, hogy a Linux még nem készült fel a Nagy Vasakra, meglepetésre számítson.

Mivel gyorsan növekedtünk, az NFS alrendszer kezdett problémássá válni. Egészen pontosan néhány gép összeomlott egy bizonyos NFS kiszolgáló–ügyfél-változat használata esetén. Bár tapasztalataink szerint az NFS-ügyfelek igen erőteljesek, a jelenlegi Linux NFS szolgáltatásokon azért van még mit csiszolni. A leggyorsabb NFS kiszolgálónk, egy IBM x342 (2xP3-1.24, 2GB RAM) sem volt képes

GSC MySQL LIMS

2,1 millió minőségi bázispárt tartalmazó 3250 szekvenciát gyűjtöttünk be, amelyeket a kezdeti vázlat összeszerkesztéséhez továbbítottunk. Ez körülbelül 70× fedi le redundáns módon a genomot. A WGS során általában csak 10×-es ismétléssel dolgozunk, de számunkra az időtényező volt a legfontosabb, így el akartuk kerülni az első sorrend-meghatározási körben nem teljesen lefedett részek miatt bekövetkező késlekedést.

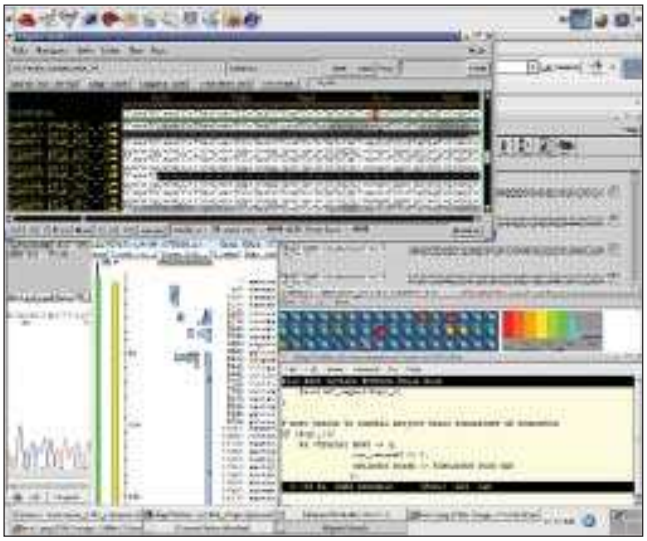
```
SELECT
SUM(Sequence_Length) AS bp_tot,
AVG(Quality_Length) AS bpq_avg,
SUM(Quality_Length) AS bp_qual_tot,
COUNT(Well) AS reads,
Sequence_DateTime AS date,
Equipment_Name AS equip
FROM
Equipment, Clone_Sequence, Sequence_Batch, Sequence,
Plate, Library, Project
WHERE FK_Sequence_Batch__ID=Sequence_Batch_ID AND
FK_Plate__ID=Plate_ID AND
FK_Library__Name=Library_Name AND
FK_Equipment__ID=Equipment_ID AND
FK_Project__ID=Project_ID AND
FK_Sequence__ID=Sequence_ID AND
Sequence_Subdirectory like "SARS2%" AND
Quality_Length > 100 AND
Sequence_DateTime < "20030413"
GROUP BY Sequence_ID ORDER BY Sequence_DateTime;

bp_tot bpq_avg  bp_tot reads  date              equip
437256 612.6399 205847 336 2003-04-11 21:07:06 SARS212.B21 D3730-3
412366 752.1074 245187 326 2003-04-11 22:15:34 SARS213.B21 D3730-1
269456 639.1926 225635 353 2003-04-11 22:22:34 SARS215.B21 D3700-6
130525 715.5060 118774 166 2003-04-11 22:25:44 SARS216.B21 D3700-5
282490 682.6311 249843 366 2003-04-11 22:27:14 SARS215.BR D3700-4
310119 612.7601 212015 346 2003-04-11 22:31:56 SARS213.BR D3700-1
182573 681.4975 136981 201 2003-04-11 22:36:40 SARS216.BR D3700-3
301471 642.2273 226064 352 2003-04-12 01:58:16 SARS212.BR D3700-2
401595 690.5204 220276 319 2003-04-12 05:13:26 SARS211.BR D3730-3
460100 642.0468 219580 342 2003-04-12 06:20:52 SARS214.BR D3730-2
182360 471.7832  67465 143 2003-04-12 07:14:44 SARS214.B21 D3730-1
```

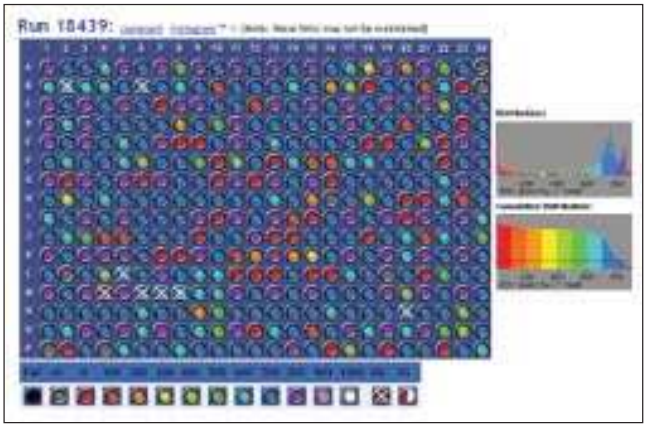
7. kép Számítási és tárolási hátér: 1. A kezdő lépések 2001 januárjában az x330 kiépítésében, *2.* 84 darab x330-as csomópontot és *3.* NetApp FAS960 filer és két IBM 3583 LTO egy x342-esen futó Veritasszal felügyelve és irányítva

4000-6000 NFS művelet/másodperc értéknél többre, különösen, ha nagy mennyiségű olvasási, illetve írási műveletet kapott a fürttől. A teljesítménykorlátok kezelésére beszereztünk egy NetApp FAS960 Filert (*7. kép*). 10 TB nyers tárkapacitása mellett (5×14×144 GB) a filer elérte a 30 000 NFS művelet/másodperc teljesítményt. Az NFS gondok ellenére az eredeti VAR900 termelési kiszolgálónk (*2. kép*) a megbízhatóság mintaképe lett, és 2002 februárjában elérte a 394 napos működési időt, amikor is fejlesztés miatt újraindítottuk.

8. kép A BLAST-lekérdezés legjobb találata



9. kép Nukleotidsorrend vizsgálata Linux munkakörnyezetben



10. kép Az egyik SARS-lemez szekvenciáinak minőségi értékei

Az első Tor2/SARS-adatok 2003. április 11-én, pénteken kerültek információs csoportunkhoz elemzésre. A szekvenálás helyességének igazolására ellenőriztük az esetleges szennyezettséget. BLAST keresés segítségével meg tudtuk határozni, hogy milyen legközelebbi egyezést mutat nyilvános fehérje- (proteom) és nukleinsav- (genom) adatbázisokkal. Nagy megkönnyebbülésünkre a legjobb találatot a szarvasmarha coronavírus adta (*8. kép*), ami azt jelentette, hogy valóban olyasmit szekvenáltunk, aminek köze van a coronavírusokhoz. Ezeknek a vírusoknak A- (adenin-) sorozattal zárul a sorrendje, így amikor megláttuk, hogy bizonyos leolvasások poly-A „farokban” végződnek, bíztunk benne, hogy azok a genom egyik végét jelentették.

A SARS adatainak összeállítására és vizsgálatára az x330-asokat és az x440-est használtuk. A genom nem túl nagy, így az összeállítása egyetlen CPU-n nem vett többet igénybe 15 percnél. Összehasonlításképpen, az emberi genom első nyilvánosságra hozott sorrendje 300 000-szer volt nagyobb a Tor2/SARS méreténél, és az összeállítása négy napon keresztül folyt az UCSC-nél, egy százprocesszoros Linux-fürtön. 2003. április 12-én szombat éjjel 2:25-órakor befejeztük a Tor2/SARS összeállításának hetedik ismétlését, és ezt az állapotot fogadtuk el az első érvényes vázlatként. Ezt importáltuk az AceDB-be, hogy lássuk, mennyire illeszkedik a már ismert proteinkészletekhez (*9. kép*). A szombatot az összeállításunk kiértékelésével töltöttük, amit aztán egy nappal később az x440-esünk saját, Zope/Plone alapú CMS rendszert futtató nyilvános webkiszolgálójára tettünk fel.

Összegzés

A Tor2/SARS genomját egy negyedik, újfajta coronavírus-csoport tagjaként azonosítottuk, ami információt szolgáltat diagnózisteszthez, a jövőben pedig esetleges terápia kifejlesztéséhez, beleértve oltóanyag előállítását is. A Linux lehetővé tette, hogy célunkat úgy érjük el, hogy közben nem kell egy vagyont költenünk eszközökre és programokra. Tömegcikként gyártott alkatrészek beépítésével elkerülhettük a hosszú megvalósulási idő miatt bekövetkező értékcsökkenést. Figyelni fogjuk az újonnan felmerülő hibákat, mindeközben MySQL adatbázisunk tárt kapukkal várja az új szekvenciákat.

Köszönetnyilvánítás

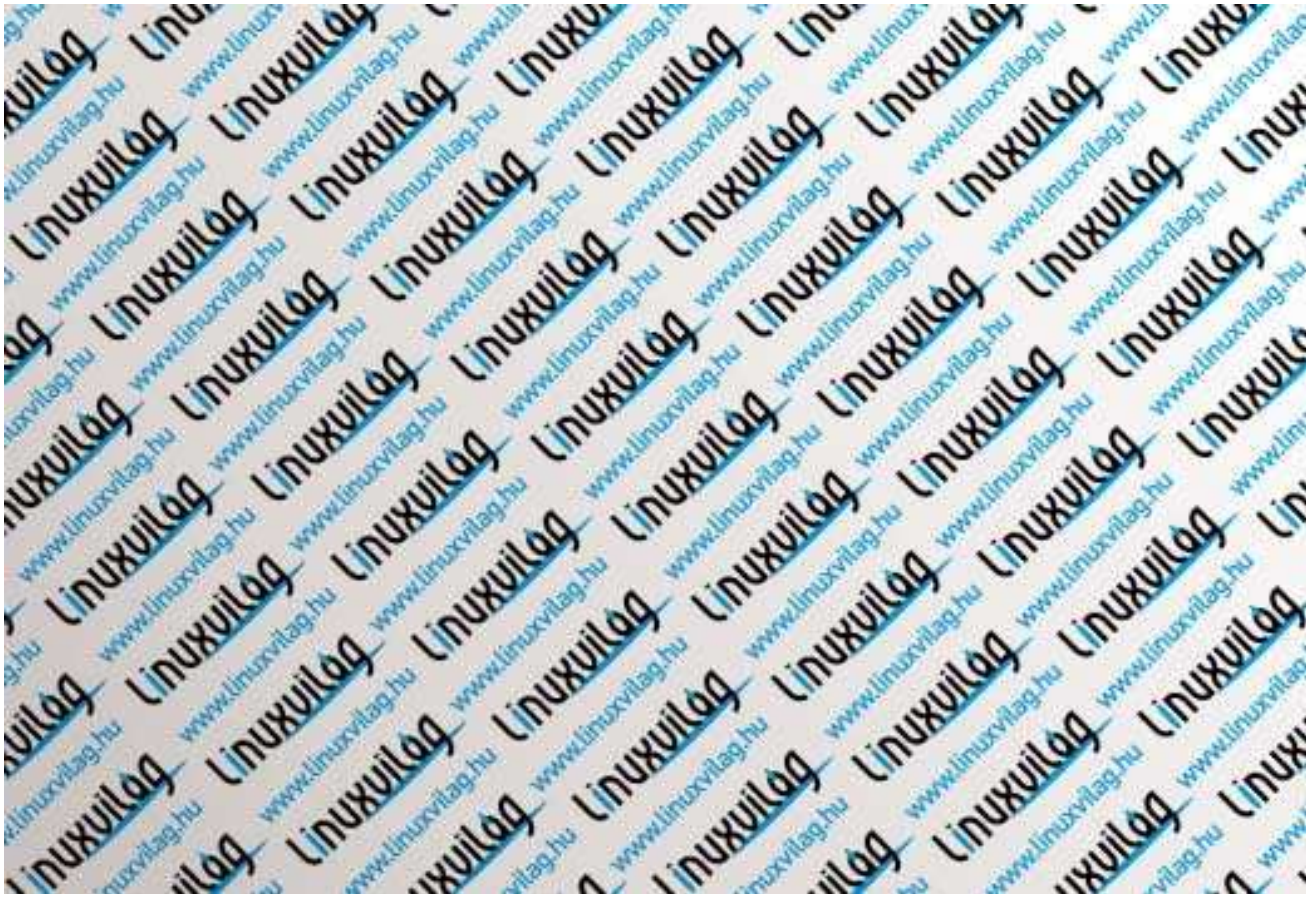
A szerzők szeretnének köszönetet mondani ***Marco Marra, Steven Jones, Caroline Astell, Rob Holt, Angela Brooks-Wilson, Jas Khattra, Jennifer Asano, Sarah Barber, Susanna Chan, Allison Cloutier, Sean Coughlin, Doug Freeman, Noreen Girn, Obi Griffith, Steve Leach, Mike Mayo, Helen McDonald, Steven Montgomery, Pawan Pandoh, Anca Petrescu, Gord Robertson, Jacquie Schein, Asim Siddiqui, Duane Smailus, Jeff Stott*** és ***George Yang*** hölgyeknek és uraknak tudományos szaktudásukért, valamint laboratóriumi és bioinformatikai erőfeszítéseikért. Szeretnénk köszönetet mondani ***Kirk Schoeffelnek***, ***Mark Mayonak*** és ***Bernard Linek*** rendszerfelügyeleti tanácsaiért.

A cikkhez tartozó Kapcsolódó címek az 54. CD Magazin/SARS könyvtárában találhatóak.

Linux Journal 2003. november, 115. szám

Martin Krzywinski (martink@bcgsc.ca)
Bioinformatikai kutató a kanadai Michael Smith Genomtudományi Központban. Idejét fizikai hozzárendelés és adatfeldolgozás-automatizálási kérdések megoldásával tölti a Perl nyelv segítségével.

Yaron Butterfield (ybutterf@bcgsc.ca)
A szekvenáló bioinformatikai csoportot vezeti a kanadai Michael Smith Genomtudományi Központban.



A TALOSS program

A Naval Undersea Warfare Center (a Tengerészet Tengeralatti Hadviselés Központja) új projektje azt vizsgálja, hogy egy összevont háromdimenziós kijelző segíti-e az amerikai tengeralattjárók parancsnokló tisztjeit a jobb döntések gyorsabb meghozatalában.

Az amerikai hadsereg átvette a hálózatközpontú (net-centric) működés elvét, ennek a célja a parancsmeghozatal folyamatának gyorsítása. E módszer használatával egy-egy parancs gyors megszületése három része bontható:

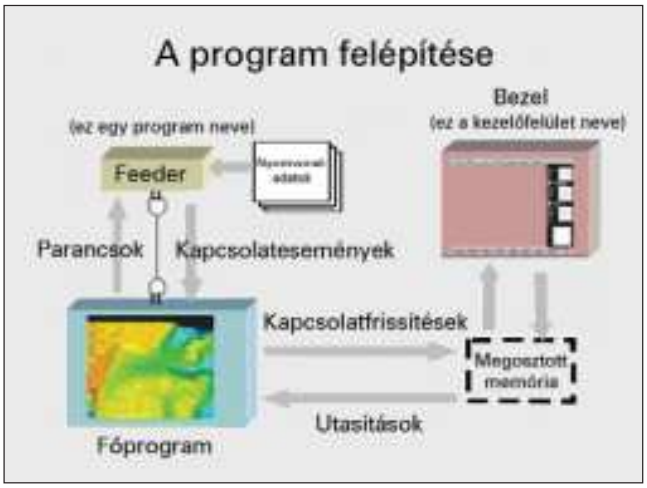
1. a katonai erő információs fölényt ér el, abszolút jobban tájékozott, vagyis felfogja, megérti a csatatéren lévő helyzetet, nem csak nyers adatokkal rendelkezik;
2. az erők gyorsan, pontosan és nagy hatótávolsággal tevékenykednek, a fölényt nem kizárólag az erők összpontosítása, hanem a nagyobb hatékonyság révén érve el;
3. ezek eredménye az ellenség lehetséges tevékenységének kizárása és a lökésszerű, szorosan egymáshoz kapcsolódó események.

Még egy említésre méltó dolog tartozik a hatékonysághoz: az elosztott katonai harci rendszerek harcképesen egyesíthetők, hogy a harci erők minden pillanatban a lehető legjobban naprakészek legyenek a harctéri helyzet tekintetében. A hálózatközpontú hadviseléshez egy általános hadműveleti, illetve harcászati képre van szükség, vagyis minden felületre kell egy állandó és megbízható általános harcászati kép (CTP – Common Tactical Picture), hogy a harcierők a harctéri helyzet lehető legjobb ismeretét érjék el a többszintű hadviselésben.
A kihívás a gyors parancsadás megvalósításában a fürge, pontos helyzetismeret kifejlesztése és a teljes harctér állapotának megértése. Hagyományosan és jelenleg is a döntéshozók több kétdimenziós kijelző és nyomtatott papírok adatainak az összevetésével létrehozzák a harctér szellemi (mentális) modelljét. A hálózatközpontú vagy osztott környezet a harctér megjelenítésének új megközelítését igényli – mind részletességét, mind megjelenítés módját illetően.
A TALOSS (Three-dimensional Advanced Localization Observation Submarine Software, azaz háromdimenziós fejlett felderítő és megfigyelő tengeralattjáró program) rendszer feladata, hogy az összetett adatok gyors, illetve pontos összevetését, feldolgozását tegye lehetővé a tengeralatti harctéren. A TALOSS képes egy általános tengeralatti harcászati képet létrehozni, amelyen láthatók a becsült fenyegetettségi zónák, az érzékelők kapcsolatkövetései, valamint a saját hajó helyzete és iránya, mindez kombinált navigációs/topográfiai/batimetrikus (mélytengermérési) adatokkal megjelenítve. Feltevések szerint ez az egyesített tengeralatti kép gyorsabb és pontosabb döntéshozatalt tesz lehetővé, valamint tökéletesebb tervezési és döntési segítséget nyújt.

Általános felépítés

A TALOSS együttműködik a Red Hat 7.0-9.0 és Slackware 9 terjesztésű Linux-rendszerekkel. Több okból választották a Linuxot operációs rendszerként:

1. együttműködik (compatibility) a jelenlegi és a jövőbeni tengeralattjáró harci rendszereivel;



1. ábra A TALOSS alapfelépítése

2. ez egy általános Unix operációs rendszer, ami azt jelenti, hogy a Linux alatt létrehozott programok és parancsfájlok könnyedén átvihetők más Unix operációs rendszerekre, például a HP-UX-ra és az Irixra;
3. ez nyílt forrású operációs rendszer, nagy felhasználói közösséggel, amelynek tagjai közösen látják el a rendszer karbantartását és javítását (optimalized).

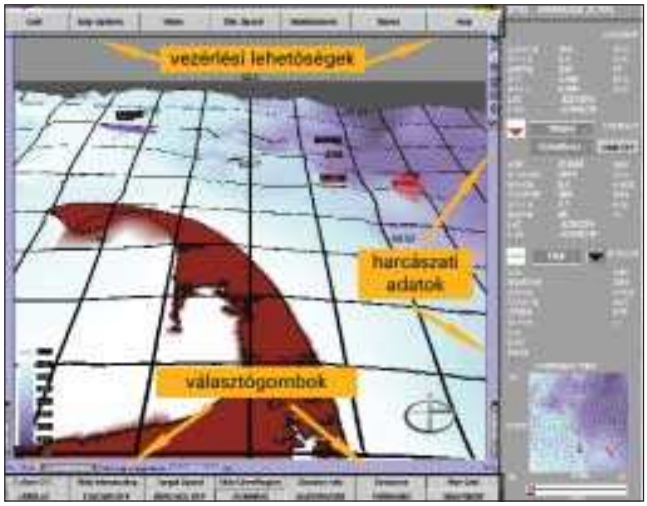
A TALOSS három fő modulból áll, ez a Feeder, a Beezel és a 3D-s kijelző. Az alapprogram felépítésének váza az *1. ábrán* látható.

Harcászati adatbevitel

A Feeder program TPC/IP-foglalaton keresztül olvassa be és küldi a fő kijelzőre a tengeralattjáró harci irányító rendszer (CCS – Combat Control System) adatait. A program közvetlenül a harci adatbázishoz kapcsolható, vagy előzőleg rögzített, illetve szemléltető (demonstration) adatokat futtathat ASCII bemeneti fájlból. A Feeder rugalmas modul, párhuzamosan könnyen át lehet alakítani többféle adatbázis tartalmának a beillesztésére. Ennek a rugalmasságnak köszönhetően a TALOSS rendszer együttműködik a nem tengeralattjáróval kapcsolatos alkalmazásokkal is, például óceanográfiai, illetve topográfiai 3D-s térképekkel, 3D-s sebességmező- és 3D-s radar-, illetve szonártérképekkel, valamint minden olyan alkalmazással, amelyben az objektumok 3D-s környezetben mozognak.

Szimulációvezérlés

A Beezel egy olyan információs és vezérlő grafikus kezelőfelület (GUI), amely Fast Light Tool Kit (FLTK) felhasználásával készült. Ez vezérli a 3D-s fő alkalmazást és korlátozott mértékben a Feeder programot. Ezenkívül rendszervezérlő feladatához kapcsolódóan a rendszer állapotát is jelzi. A rendszerálla-



2. ábra A teljes TALOSS-kijelző

potot egy ***Open Inventor*** 3D-s ablakban jeleníti meg, megmutatva a harctér felülnézeti képét, középen a saját hajóval. Ez alapjában véve a 3D-s helyszín 2D-s nézete, amely tájékozódási pontul szolgál a felhasználónak, hogy a 3D-s színen könnyen megállapítható legyen a saját hajó helyzete és iránya. A Beezel és a főprogram a megosztott memórián keresztül cserél adatokat. A *2. ábrán* egy teljes TALOSS-képernyő látható a Beezel FLTK vezérlőelemeivel, amelyek az Open Inventor 3D-s képernyőjéhez vannak kapcsolva.

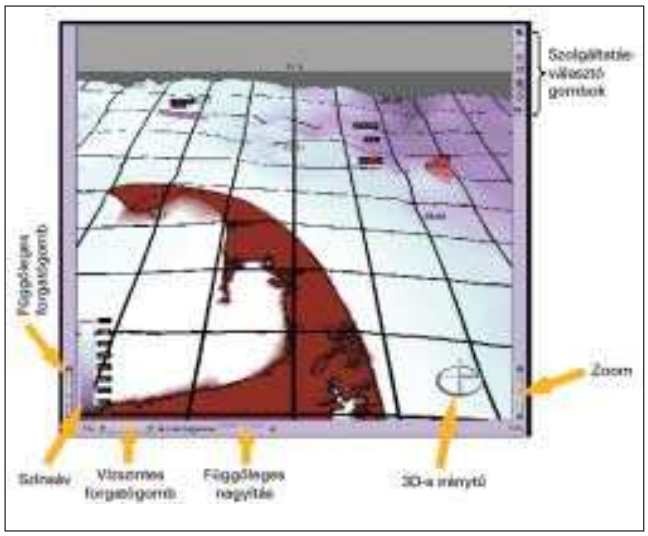
A 2. ábrán vizsgáljuk meg a a Bezel tetején lévő lenyíló menüt. Ennek segítségével lehet elérni a TALOSS alapszolgáltatásait: a kilépést, a nézetet, a térkép színének változtatását, az uralkodó mélységhatárokat és így tovább. Hasonlóan a képernyő alján két sor váltógomb húzódik, amelyekkel a 3D-s képernyőn látható tengeralatti harctér harci adatainak a megjelenítését lehet szabályozni.

A Bezel jobb oldala különbözik a felső és az alsó résztől, mivel innen az éppen szükséges taktikai információk olvashatók le:

1. saját hajó-, célpont- és fegyveradatok;
2. a cél- és érzékelőválasztás állapota;
3. numerikus célpont-elkülönítési területadatok.

Ezenkívül, mivel a teljes rendszer egy közös műveleti órához van hangolva, ami a haditengerészet saját daytime group (DTG) formátumában dolgozik, ez is megjelenítésre kerül Bezel jobb felső részén. A *3. ábra* a Bezel jobb oldalának jellemző állapotát mutatja meg.

3. ábra A Bezel jobb oldali tulajdonságablaka



4. ábra A 3D-s kijelző tulajdonságai

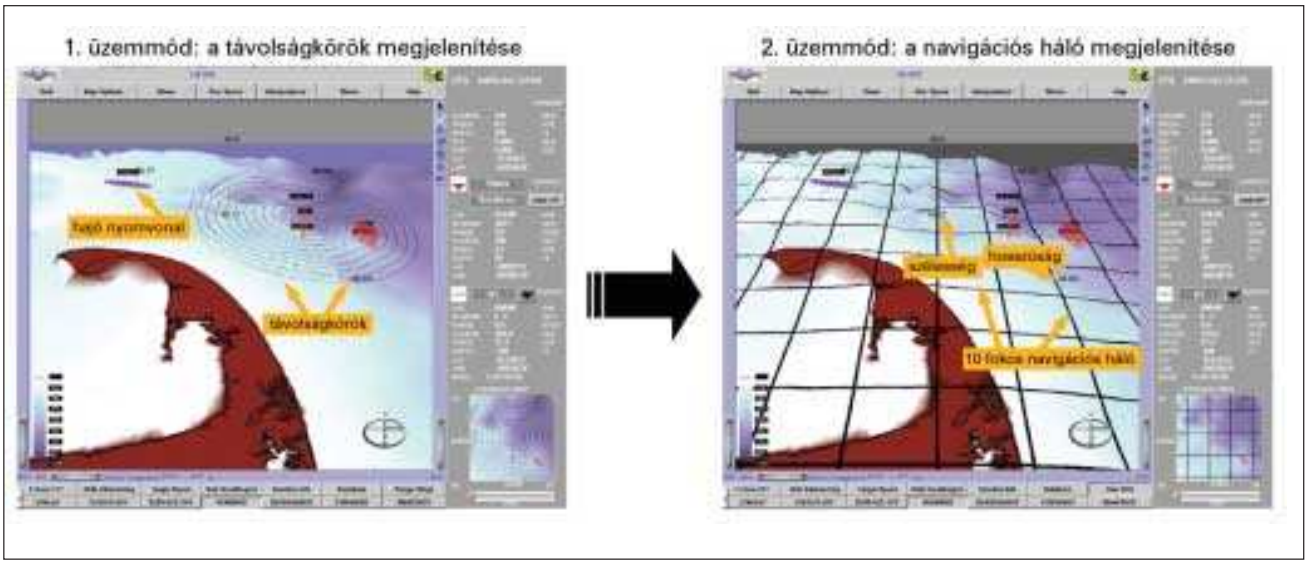
3D-s helyszínkezelés

A 3D-s tengeralatti harctérkijelző a harcászati információkat a Feeder programtól kapja. Ezek olyan térképészeti és navigációs adatok a Digital Nautical Chart (digitális hajózási térkép) adatbázisból, amit indításkor töltenek fel, és a harcászati helyzet előrehaladtával folyamatosan frissítik. Minden navigációs adat Open Inventory bináris formátumban előre el van készítve, és ezeket Navigation Tile-nak (navigációs mozaiknak) nevezzük. A főprogram az összes harcászati és navigációs adatot egy átfogó 3D-s képpé dolgozza össze a felületfüggetlen helyszínrajzkészítő, az Open Inventor segítségével.

A 3D-s képernyő bal alsó sarkában három vezérlőelem található. A ***Rotx*** tekerőgomb a helyszínt elforgatja egy, a képernyőn vízszintesen keresztbe futó, képzeletbeli x tengely körül. A ***Roty*** tekerőgomb a helyszínt a képernyőn keresztbefutó képzeletbeli y tengely körül forgatja el. A függőleges nagyítás csúszka a helyszín képe által átfogott mélységtartományt módosítja. A kezelőelemek használatához a kurzort egyszerűen föléjük kell vinni, a bal egérgombot le kell nyomni és nyomva kell tartani, miközben az egeret a kívánt irányba húzzuk.

A 3D-s kijelző jobb alsó sarkában található a ***Zoom*** tekerőgomb. A zoommal csak a tengerfenékig lehet közelíteni.

A megjelenítést közvetlenül módosító kezelőszerveken kívül a 3D-s képernyő hét nyomógombot tartalmaz. Ezekkel lehet például a kapcsolatot kiválasztani, a helyszínképet kezelni, alapnézetet (home) beállítani és a rácsmegjelenítést bekapcsolni. Érdemes még megemlíteni néhány egyéb 3D-s szolgáltatást is: a lebegő 3D-s iránytűt, amely egyszerre jelzi az irányt és a



5. ábra Navigációs és nyomkövető adatok

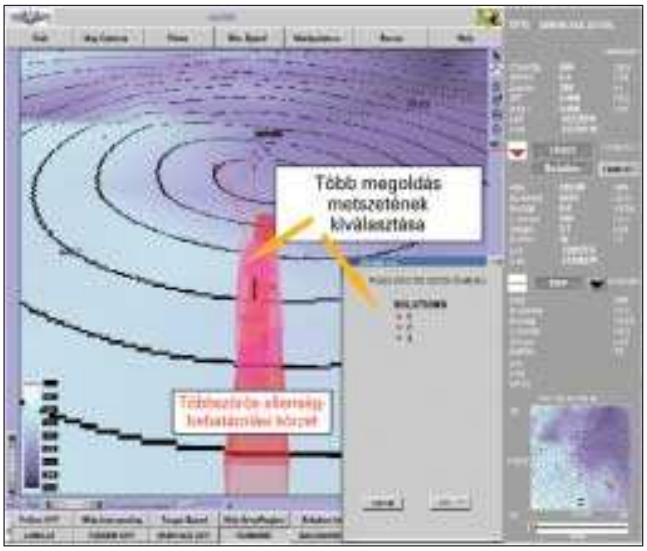
helyszínkép tájolását, valamint a színskálát, amely az árnyalással jelzett mélységadatok értékeléséhez nyújt segítséget.
A mélységárnyalás módja a színtérképtől és a beállított uralkodó mélységtől függ (lásd a *4. ábrán*).

Hadművelet

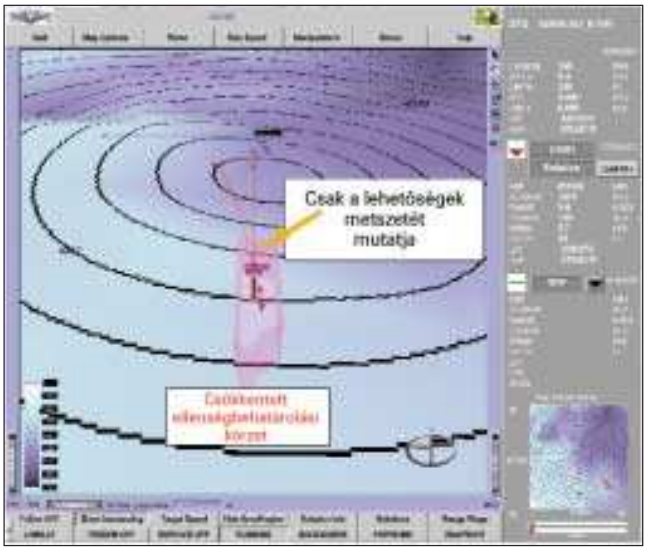
A harcászati kijelző szerepe, hogy az összes objektum stb. helyzetét a kezelő számára a saját pozíciójához képest érzékelhetővé tegye. Ezt a TALOSS a saját hajó köré rajzolt koncentrikus körökkel éri el, ezek segítségével a kezelő vizuálisan azonnal meg tudja állapítani egy adott objektumnak a saját hajótól való távolságát. Ez az adat különösen az ütközések elkerülése és a fenyegetettség felmérése miatt életbevágóan fontos. Ezek a körök a 3D-s képernyőn és a Bezel felülnézeti képén egyaránt láthatók.
Minden harcászati vagy állapothelyzet-megjelenítés alapvető összetevője a földrajzi helyzet pontos ismerete. A TALOSS kétféle módon támogatja a navigációs adatokat:

1. egy 10 fokos beosztású navigációs rács használatával. Mivel a szélességi vonalak közötti távolság mindig állandó, a függőleges vonalak között 10 mérföld távolság van. A hosszúsági vonalak távolsága a szélességi pozíciótól függően változik, de a mérsékelt övben ez is kb. 10 mérföld.
2. A hosszúsági és szélességi vonalakon kívül egy alfanumerikus érték is látható a 3D-s térképen, ami a hosszúságot és a szélességet jelzi. Ezek az értékek és a rácsvonalak a Digital Nautical Chart (digitális hajózási térkép) adatbázisából származnak. A navigációs rács felváltva jelenik meg a távolságkörökkel, míg az alfanumerikus kiírás mindig látható. Az *5. ábrán* a 3D-s kijelző az összes navigációs információval látható.

A helyzetismereti megjelenítés legfontosabb összetevője az a képesség, hogy minden mozgó objektumot vizuálisan követ és egységesít. A TALOSS-ban térbeli vonalak jelzik minden ismert objektum becsült nyomvonalát. A vonalak színe jelzi az objektumok hovatartozását, ami lehet baráti (kék), ellenséges (vörös) vagy semleges (sárga). Mivel a TALOSS jelenlegi változata tengeralattjárón történő használatra készült, megjeleníti az ellenséges és saját fegyverek nyomvonalát is, narancsszínnel, illetve zölddel. Ez a színséma könnyen módosítható más célú felhasz-



6. ábra 3D-s behatárolási körzet



7. ábra 3D-s metszet

A tengerészeti átrendezési ütemterv honlapja

nálásra. Az 5. ábrán megfigyelhető néhány kapcsolat nyomvonala, valamint a saját hajó helyzete. Minden kapcsolat nyomvonala a Bezel kapcsolatablakában megadottak szerint fel van címkézve. A nyomkövetésen kívül a TALOSS a lehetséges veszélyzónákat 3D-s behatárolási régióként folyamatosan nyilvántartja. Ugyanazt a vörös, kék és sárga színsémát használja a megkülönböztetésükre. A *6. ábrán* egy jellegzetes 3D-s behatárolási régió látható. Ezek a régiók összetett térbeli testekként láthatók. Kiválasztásuk a Bezellel történik, ami egyaránt lehetővé teszi a régió kijelölését és egy adott régióhoz tartozó kapcsolat kiválasztását. A behatárolási körzetek növekedhetnek és metszhetik egymást, így imitálva az idő múlásával a hajók mozgását. A körzetek növekedése minden lehetséges helyet ábrázol, ahol a hajó a behatárolási körzet térfogatán belül előfordulhat, ha netán megszakad a pontos érzékelőkapcsolat. Mikor egy érzékelő újból megtalálja a hajót, a friss adatot egy új 3D-s behatárolási térfogat formájában képezi le, amit ki lehet vonni az előzőleg megnövelt 3D-s térfogatból, hogy egy jócskán csökkentett közös térfogatot kapjunk (lásd a *7. ábrát*). Ez a közös térfogat tartalmazza a legnagyobb valószínűséggel a számunkra érdekes hajót. Egy katonai harcászati rendszer célja, hogy gyorsan lokalizálja a fenyegetést jelentő térfogatot, hogy a megfelelő ellenintézkedést villámgyorsan meg lehessen hozni. Más szóval: harcolj vagy tűnj el. A két vagy több nem folytonos térfogat metszetét elkészítő, úttörő jellegű programot az Arizonai Állami Egyetemen fejlesztették ki a TALOSS-hoz. A Linux programfejlesztéshez történő használatának egyik legfontosabb előnye, hogy az egyes részterületek kutatásait az egyetemeken vagy más nyílt forrású helyeken olcsón el lehet végezni. A Linux használata azt jelenti, hogy a résztvevő kutatópartnereknek nem kell olyan drága fejlesztői felületeket beszerezni, például Hewlett-Packard TAC vagy Silicon Graphics munkaállomásokat. Ehelyett olcsó, Linux alapú PC-ken tudnak olyan kódot létrehozni, amelyek könnyen beilleszthetők a harcászati rendszerbe. Az amerikai kormány szorgalmazza a kereskedelmi forgalomban bárki által beszerezhető eszközök használatát mind a fejlesztéshez, mind a harcrendbe állításhoz, hiszen egyrészt ez költségkímélő megoldás, másrészt így hosszú távon biztosított a karbantartás. A Linux használata a fejlesztés során és a katonai rendszerben üzemi környezetben jó példa a kereskedelmi forgalomban kapható eszközök alkalmazására.

KAPCSOLÓDÓ CÍMEK

„Tengerészeti átrendezési ütemterv, tengeri csapás, tengeri pajzs, tengeri medence”, Tengerészeti Minisztérium, Washington, DC, 2002 (nyílt). A ➲ http://spica.gl.nps.navy.mil/Orarchives/SEA-TRIAL/NavalTransformRdMap.pdf címen érhető el.

K. Lima „Megjelenítés a többszintű hadviselés tervezésében és végrehajtásában”, ONR parancsnoki, valamint irányító és harci rendszerek találkozó 2002, Arlington, Virginia, 2002. április 23–25. (nyílt)

G. M. Nielson és *G. L. Graf*: „Térfogat modellezési technikák alkalmazása a tengerészeti alkalmazások dinamikus behatárolási körzeteiben”, Interim Progress, ONR Grant N0014-02-1-0287, Arizonai Állami Egyetem, Phoenix, Arizona, 2002. május 15. (nyílt).

R. Shell, L. Mathews, K. Lima, R. King és *F. Das Neves* „Tengeralatti vezetési és irányítási megjelenítés”, Elhangzott 2001. július 22–26-án a 7. éves egyesített légi- és űrfegyver-rendszerek, érzékelők és szimulációk szimpózium és kiállítás, 2001. San Diego, California.

Összegzés

Egy rugalmas, modulokból felépített 3D-s adategyesítéses (data fusion) megjelenítő rendszer katonai és civil célokra széles körben, egyaránt alkalmazható. A Rhode Island-i Newportban lévő Tengerészet Tengeralatti Hadviselés Központja a Haditengerészeti Kutatóiroda támogatásával létrehozott egy rendszert a tengeralattjáró tenger alatti harcterének megjelenítésére és egyesítésére. A TALOSS elnevezésű rendszer többféle adatbázis tartalmának összevonására képes, civilére és katonaiéra egyaránt. Mivel a program modulokból épül fel és Linux alatt íródott, megvan a lehetőség, hogy nyílt forrású rendszerré és adatfúziós motorrá alakítsuk át. A projekt végső célja egy teljesen modularizált TALOSS-eszközkészlet kifejlesztése, amelynek nem titkos részei civil célokra is felhasználhatók lennének a nyílt forrással. Létrehozásának két fő motivációja létezik:

1. a program nyilvános alapon készült, és amennyiben a nemzetbiztonságot nem veszélyezteti, nyilvánosnak is kell maradnia;
2. remélhető, hogy a program nem titkos részeinek elérhetővé tételével a nyílt forrás közössége által végzett teljesítménynövelő javítások beépíthetők a titkosított részbe is, növelve annak hatékonyságát.

Linux Journal 2003. november, 115. szám

Douglas B. Maxwell Jelenleg a Tengerészeti Tengeralatti Hadviselés Központja Fegyverek és Ellenintézkedések ágazatának tagja (2213-as kód). 2001-ben kapta meg tudományos fokozatát a luisianai műszaki egyetemen.

Richard Shell
Elektronikai és informatikai mérnök a Rhode Island-i Newportban lévő Tengerészet Tengeralatti Hadviselés Központjában.

Osztott rendszerű hasítótáblák (1. rész)

Az egyenrangú hálózatok világában az osztott rendszerű hasítótáblák (Distributed Hash Tables, DHT) forradalmi újítást jelentettek. Tanuljuk meg, hogyan írhatunk olyan alkalmazást, amely lehetővé teszi, hogy mindenki azonos adatokon dolgozzon.

Az első nemzedékbeli egyenrangú hálózatok (peer-to-peer) áttekinthetetlen, ötletszerűen felépített szerkezete helyett ma már egyre gyakrabban alkalmaznak kiváló teljesítményű, hasznos tulajdonságokkal rendelkező, tervezett topológiakészleteket.

Számos olyan kísérleti DHT-t készítettek különböző egyetemeken, amelyeket a nyílt forráskód közössége felkarolt és megvalósított. Létezik néhány kereskedelmi alkalmazás is, de jelenleg egyik sem érhető el SDK (programfejlesztő készlet) formában, hanem valamelyik boltokban kapható termékbe építve találjuk meg őket. Minden DHT-megoldást elképzelhetünk minden más megoldástól eltérő, önálló entitásként is. Így tulajdonképpen valamennyi létező megoldást befoglalhatnánk egy többdimenziós mátrixba. Kiválasztunk egyet, elvégzünk néhány módosítást, és máris egy másiknál járunk. A már létező kutatói DHT-k, amilyen a Chord, Kademlia vagy a Pastry, így aztán kiváló kiindulási alapként szolgálnak saját fejlesztéseinkhez.

A DHT-k lényegében egy hasítótábla feladatát látják el: kulcs és érték párokat tárolhatunk bennük. Ha ismerjük a kulcsot, megkaphatjuk az értéket. Az értékeknek nem feltétlenül kell a lemezen létezniük, de DHT rendszerünket természetesen valamilyen állandó hasítótáblára alapozhatjuk, például a Berkeley DB-re; ami egyébként ténylegesen el is készült.

A DHT rendszerekben az a különleges, hogy a tároló lekérdezései több gép között oszlanak meg. A már létező elsődleges-másodlagos kiszolgáló alapú adatbázis-szerkezetekkel ellentétben itt minden csomópont egyenértékű és szabadon csatlakozhat, illetve hagyhatja el a hálózatot. Annak ellenére, hogy a hálózat tagjai látszólag áttekinthetetlen módon, véletlenszerűen változnak, a DHT-k igen meggyőző teljesítményt mutatnak.

Kezdjük a DHT-tervezés rejtelmeinek felfedezését egy egyszerű körkörös, kétszeresen láncolt listával. A lista minden egyes csomópontja egy-egy gépet jelent a hálózaton. Minden csomópont az előtte és az utána következő elemről is tárol adatot, mégpedig az adott gép címét. Meg kell határoznunk valamiféle sorrendet, ezért minden csomópontnál megadjuk, hogy melyik lesz a „következő” csomópont. A Chord DHT által alkalmazott módszer szerint a következő csomópontot ilyen módon kapjuk meg: minden egyes csomóponthoz rendeljünk egyedi, k bites azonosítót. A gyűrű elemeit olyan módon rendezzük, hogy az azonosítók óramutató járásával megegyező irányban növekedjenek. Minden csomópont számára az lesz a következő, amelyiknek az azonosítója órairányban a legkisebb távolságra áll tőle. A legtöbb csomópont esetében ez az a csomópont lesz, amelyik a legkisebb, ugyanakkor nagyobb azonosítóval rendelkezik az adott csomóponthoz képest.

Ez alól csak a legnagyobb azonosítójú csomópont a kivétel, ennek az utódja ugyanis éppen a legkisebb azonosítóval rendelkezik. Ezt a távolságmértéket a távolságmódszer valamivel helyesebben adja meg (lásd az *1. listát*).

Minden egyes csomópont önmagában egy szabványos hasítótábla. Ha egy értékre van szükségünk, mindössze annyi a feladatunk, hogy rátaláljunk a megfelelő csomópontra, innen már a hagyományos hasítótábla-keresést vagy -tárolást kell csak elvégeznünk. Az adott kulcshoz tartozó csomópont keresésének egyik egyszerű (és a Chord által is használt) módszere, ha ugyanazt tesszük, mint az adott azonosító leszármazottainak a keresésénél. Először is vegyük a kulcsot, majd egy kereső alapján rendeljünk hozzá valamilyen pontosan k bites másik kulcsot. Tekintsük ezt a számot a csomópont-azonosítónak, majd állapítsuk meg, hogy melyik csomópont lesz a leszárma-

1. lista A ringDistance.py

```
# Ez az órajárással megegyező gyűrű
# távolságfüggvénye. A globálisan megadott k
# értéktől, a kulcsmérettől függ a legnagyobb
# lehetséges csomópont-azonosító 2**k.
def distance(a, b):
    if a==b:
        return 0
    elif a<b:
        return b-a;
    else:
        return (2**k)+(b-a);
```

2. lista A findNode.py

```
# A kezdő csomóponttól indulva keressük meg
# a célkulcsért felelős csomópontot.
def findNode(start, key):
    current=start
    while distance(current.id, key)
    ➥> distance(current.next.id, key):
        current=current.next
    return current

# Keresd a megfelelő csomópontot és kérd
# le a kulcshoz tartozó értéket.
def lookup(start, key):
    node=findNode(start, key)
    return node.data[key]

# Keresd a megfelelő csomópontot és tárold
# az értéket a kulccsal.
def store(start, key, value):
    node=findNode(start, key)
    node.data[key]=value
```

3. lista Az update.py

```
def update(node):
    for x in range(k):
        oldEntry=node.finger[x]
        node.finger[x]=findNode(oldEntry,
(node.id+(2**x)) % (2**k))
```

4. lista A finger-lookup.py

```
def findFinger(node, key):
    current=node
    for x in range(k):
        if distance(current.id, key)
        ➥> distance(node.finger[x].id, key):
            current=node.finger[x]
    return current

def lookup(start, key):
    current=findFinger(start, key)
    next=findFinger(current, key)
    while distance(current.id, key)
    ➥> distance(next.id, key):
        current=next
        next=findFinger(current, key)
    return current
```

zottja, azaz a gyűrű bármely pontjáról indulva az óramutató járásával megegyező irányba gyalogoljunk, addig, amíg rá nem akadunk arra csomópontra, amelynek az azonosítója a legközelebb van, de már nagyobb, mint az adott szám. Az imént megtalált csomópont lesz a felelős az adott kulcs tárolásáért és visszakereséséért (lásd a 2. *listát*). A kulcs készítéséhez érdemes hasítótáblát használnunk, hiszen a hasítótábla általában egyenletes eloszlást készít, így a különféle kulcsok a hálózat csomópontjai közt egyenletesen oszlanak majd meg.

Ez a DHT-kialakítás egyszerű és egyben minimálisan szükséges is, amennyiben osztott rendszerű hasítótáblát szeretnénk létrehozni. Feltéve, hogy a rendelkezésre állással bíró csomópontok statikus hálózatát tekintjük, és az adott kulcshoz tartozó csomópontot keressük, bármilyen kulccsal és csomóponttal kezdhetünk. Fontos azonban észben tartanunk, hogy bár a példakód úgy néz ki, mintha egyszerű, kettős kapcsolatú listát kellene megjegyeznünk, ez csak egy DHT-szimuláció. Egy valódi DHT esetében minden egyes csomópont külön gép lenne, és minden egyes hívást valamiféle foglalat- (socket) protokoll segítségével kellene megvalósítani.

Ha egy kicsit hasznosabbá akarjuk formálni a modellünket, jó lenne feljegyezni, hogy mely csomópontok hagyták el a hálózatot, illetve melyek csatlakoznak hozzá akár szándékosan, akár valamilyen hiba miatt. A képesség beépítéséhez hálózatunkban be kell vezetnünk valamilyen csatlakozó-, illetve elhagyóprotokollt. A Chord csatlakozóprotokoll első lépése, hogy a hagyományos keresési szabály alapján kikeressük az új csomópont azonosítójának az utódját. Az új csomópontot e közé a leszármazott csomópont és annak elődje közé kell beszúrnunk. Az új csomópont elődje keresőszerkezetének egy részéért lesz felelős. Mivel azt szeretnénk, ha minden lekérdezés hibátlanul menne végbe, a kulcsok adott tartományát át kell másolnunk az új csomópontra, mielőtt az elődcsomópont a következő csomópont mutatóját új csomópontunkra változtatná át.

A kilépés nagyon egyszerű: a kilépő csomópont minden adatát az elődjére másolja. Az előd ezután a következő mutatóját a kilépő csomópont utódjára állítja át. A belépő és kilépő kód nagyon hasonlít a hagyományos láncolt listák törlő és beszúró utasításaihoz, kiegészítve a ki- és belépők, valamint a szomszédaik közötti adatcsere-forgalommal. Egy hagyományos láncolt listában éppen azért törlünk egy elemet, mert a benne tárolt adatot meg akarjuk szüntetni. A DHT-k esetében a csomópontok beszúrása és eltávolítása teljesen független az adatok beszúrásától és törlésétől. A DHT-csomópontokat elképzelhetjük úgy is, mint bizonyos állandó hasítótábla-megoldások időnként kiigazított vödreit (bucket), amilyeneket például a Berkeley DB is alkalmaz.

Az, hogy hálózatunkat dinamikus csomópontfelvételre és -kiadásra tettük képessé, miközben a tárolás és a lekérés továbbra is folyamatos, egyértelműen nagy fejlődés. Ugyanakkor a teljesítmény borzalmas – `O(n)` mégpedig `n/2` várható teljesítménnyel. Minden egyes közbenlévő csomópont a hálózat egy másik tagjával hoz létre kapcsolatot, amelyhez (a kiválasztott átviteltől függően) valószínűleg TCP/IP-kapcsolat szükséges. Így aztán az `n/2` csomóponton áthaladva elég lassú lehet.

Hogy ennél valamivel jobb teljesítményt érjen el, a Chord modell egy réteget vezet be, amellyel `O(log n)` teljesítmény érhető el. Ahelyett, hogy egyszerűen csak a következő csomópontot azonosító mutatót tárolnánk, minden csomópont egy `finger table`-t tartalmaz, amelyben k csomópont címe található meg. A jelenlegi csomópont azonosítója és a `finger table`-ban található csomópontok azonosítója közötti távolság exponenciálisan növekszik. Egy adott kulcshoz vezető ösvényen minden csomópont, amin csak áthaladunk, logaritmikusan közelebb lesz, mint az előző, így végül összesen `O(log n)` csomóponton haladunk keresztül.

Hogy a logaritmikus keresések jól működhessenek, az `finger table`-nek mindig frissnek kell lennie. Az elavult táblák ugyan nem rontják el a keresést, amíg minden csomópont friss „következő” mezővel rendelkezik, de a keresés csak akkor lesz logaritmikus, ha a táblák helyesen vannak felépítve. A `finger table` frissítéséhez a tábla k helyére egy-egy címet kell írnunk. Bármely x (ahol x 1-től k-ig terjedhet) helyhez tartozó `finger[x]` értéket úgy kapjuk meg, hogy vesszük a jelenlegi csomópont azonosítóját, és kikeressük az `(id+2(x-1)) mod (2k)` kulcsért felelős csomópont címét (*3. lista*). Amikor visszakeresünk, egyetlen lehetőség helyett minden ugrásnál k elemből választhatunk. Minden egyes csomópontnál, amit csak meglátogatunk, azt a bejegyzést választjuk az `finger table`-ben, amelyik a legközelebb esik a keresett elemhez (*4. lista*).

Eddig többé-kevésbé sikerült megadnunk a MIT csapat által elképzelt és kifejlesztett eredeti Chord DHT-tervet. Ez csak a DHT-jéghegy csúcsa. Rengeteg módosítást el lehet még végezni, amelyek eltérnek az eredeti Chord-leírásban bemutatott képességektől, miközben nem veszítjük el a logaritmikus teljesítményt és a Chord nyújtotta keresési biztonságot.

Az egyik ilyen képesség, ami DHT rendszerünkben jól jöhet az, ha lehetővé tesszük az `finger table` passzív frissítését, hiszen a táblák frissítéséhez időnként ismétlődő keresések szükségesek. A MIT Chordjában a tábla minden k elemére `O(log n)` csomópontot kell elérnünk, ami jelentős hálózati forgalmat is jelenthet. Előnyös lenne, ha a csomópontok fel tudnának venni más csomópontokat a saját `finger table`-jükbe, amikor valamilyen keresés miatt csatlakoznak hozzá.

5. lista Az xor-distance.py

```
def distance(a, b):
    return a^b # Python alatt, a művelet a
               # XOR b-t jelent, nem pedig
               # a szám b-edik hatványát.
```

Minthogy a keresés végrehajtása miatt a párbeszéd már egyébként is megindult, nem jelent túl sok pluszmunkát ellenőrizni, hogy a keresést végző csomópont jó jelölt-e a helyi `finger table`-ba. A Chord `finger table` hivatkozásai sajnos irány nélküliek, mivel a távolságmérték nem szimmetrikus. Egy csomópont nem feltétlenül szerepel a `finger table`-jében található csomópontok finger tábláiban.

A nehézség egyik megoldása lehet, ha a Chord moduláris összeadó távolságmértékét `XOR` alapú távmértékre cseréljük. Két csomópont, mondjuk A és B közötti távolságnak ettől kezdve a csomópont-azonosítóknak megfelelő előjel nélküli egész számok közötti `XOR` művelet eredményét tekintjük (*5. lista*). A `XOR` igen jó távolságmérték, hiszen szimmetrikus. Mivel bármely két csomópontra távolság(A, B) == távolság(B, A), ha A megtalálható B `finger table`-jében, akkor B is megtalálható A `finger table`-jében. Ez viszont azt jelenti, hogy a csomópontok frissíthetik a `finger table`-jüket az őket lekérdező csomópontok címe alapján, jelentősen csökkentve ezáltal a csomópontfrissítéssel járó hálózati forgalmat. Ez egyúttal egyszerűsíti a DHT-alkalmazás programozását, hiszen nem kell egy külön szálat fenntartani a frissítési folyamat ismételt meghívására. Ehelyett egyszerűen csak akkor frissítünk, amikor a keresési függvény meghívódik.

Az eddig bemutatott felépítéssel akadt egy kis gond, nevezetesen, hogy az adott csomóponthoz vezető ösvény igen ingatag. Ha az ösvényen található bármelyik csomópont visszautasítja az együttműködést, a keresés megakad. Bármely két csomópont között pontosan egy ösvény van, így a lerobbant csomópontok között lehetetlen megtalálni az utat. A Kademlia DHT ezt úgy oldja meg, hogy kiegészíti az `finger table`-t, úgy, hogy minden egyes bejegyzés egy helyett j csomópont-hivatkozást tartalmaz, ahol j a teljes hálózatban azonosan van meghatározva. Ettől kezdve minden ugrásra j különböző lehetőség nyílik, így körülbelül `j*log(n)` lehetséges ösvény létezik. Ennél kevesebb létezik, hiszen az ösvények összetartanak, ahogy egyre közelebb kerülnek a célhoz. Mindazonáltal a lehetséges ösvények száma valószínűleg nagyobb mint 1, ami mindenképpen fejlődés.

A Kademlia ennél is továbbmegy, és a vödrökben (bucket) a feljegyzett működési idő (uptime) alapján rendezi a csomópontokat. A régebbi csomópontok kereséseknél előnyben részesülnek, és új hivatkozások csak akkor kerülnek a rendszerbe, ha régiből már nincsen elegendő. A lekérdezések növekvő biztonsága mellett ez a megközelítés egy további előnnyel is jár: nem lehet úgy megtámadni a hálózatot, hogy gyors ütemben készítenek új csomópontokat, megpróbálván kiütni a jó elemeket – a hatást még csak észre se lehet majd venni.

Tudni kell, hogy a fent bemutatott megoldások nem egyetlen DHT-megvalósításhoz kötődnek. Lépésenként építettünk fel egy DHT-tervet az alapoktól kezdve, Chord-szerűvé formáltuk, aztán úgy módosítottuk, hogy inkább a Kademliára hasonlítson. A különféle megközelítések többé-kevésbé összekeverhetők és párosíthatók. A `finger table`-jeink vödreiben lehet egy vagy j hely, attól függően, hogy távolságmértékünkhöz moduláris összeadást vagy `XOR` műveletet használunk-e. Követhetjük mindig a legközelebbi csomópontot, vagy rangsorolhatjuk őket a fennállási idő vagy bármely más érték alapján. Meríthetünk más DHT-elképzelésekből is, például a Pastryból, OceanStore-ból vagy a Coralból. Természetesen a saját ötleteinket is használhatjuk a saját igényeink kielégítésére. A magam részéről én olyan képességekkel egészítettem ki a Chord-elképzelést, mint az anonimitás, Byzantine hibatűrő keresések, geográfiai útkeresés és az üzenetek hatékony szórása belépéskor. Érdekes megcsinálni, és sokkal könnyebb is, mint gondolnánk.

Most, hogy már tudjuk, hogyan kell saját DHT-megoldásokat készíteni, biztosan kíváncsiak vagyunk, vajon milyen őrült dolgokat lehet tenni egy ilyen kóddal. Valószínűleg rengeteg olyan DHT-alkalmazás létezik, amit még fel sem fedeztek, de már most is hallani emberekről, akik olyan feladatokon dolgoznak, mint fájlmegosztás megosztott merevlemez adatmentéshez, DNS helyettesítése egyenrangú hálózati névfeloldó rendszerrel, méretezhető csevegő és kiszolgáló nélküli játékok.

A cikkhez megpróbáltam egy mókás kis példaalkalmazást is összerakni, ami esetleg felkeltheti azoknak a figyelmét, akik látták a Linux Journal weblapján megjelent bemutatómat az egyenrangú (peer-to-peer) Superworms témáról (lásd a *Kapcsolódó címeket*). Az alkalmazás egy osztott kapupásztázó (port scanner), amely az eredményeket egy szimulált DHT-ban tárolja (6. lista; 54 CD Magazin/Hash könyvtára). Ha ténylegesen működő DHT-megoldás lenne, a parancsfájlnak néhány igazán érdekes vonása is lenne. Először is lehetővé tenné, hogy internetpásztázásának eredményeit több gép is közreadja. Ezzel a módszerrel a pásztázás többé már nem kapcsolható egyetlen géphez; továbbá elkerüli a redundáns vizsgálódást. Ha a gép már egyszer át lett vizsgálva, az eredményeket a DHT-ból kapjuk meg, elkerülve a többszörös vizsgálódást. Az adatok tárolásához nincsen szükségünk központi kiszolgálóra, az összes eredmény, illetve a tevékenység szervezése a résztvevőkön múlik. Az alkalmazás némiképpen alattomosnak tűnik, de igazság szerint a DHT könyvtár ismeretében elég nyilvánvaló volt megírása. Ugyanilyen megközelítés alapján készülhetnek el az osztott projektek is.

Kétrészes sorozatunk mostani részében áttekintettük a DHT-k mögött megbúvó elméletet. Legközelebb inkább a DHT használatának gyakorlati előnyeit vizsgáljuk meg valódi alkalmazásokban.

A cikkhez kapcsolódó listák megtalálhatóak az 54. CD Magazin/Hash könyvtárában.

Linux Journal 2003. október, 114. szám

Brandon Wiley (blanu@decentralize.org)
Az egyenrangú hálózatok betyára (peer-to-peer hacker) és a Foundation for Decentralization Research elnöke.

KAPCSOLÓDÓ CÍMEK

Achord ➲ http://thalassocracy.org/achord/achord-iptps.html
Chord ➲ http://www.pdos.lcs.mit.edu/chord
Curious Yellow ➲ http://blanu.net/curious_yellow.html
How Can You Defend against a Superworm? ➲ http://www.linuxjournal.com/article/6069
Kademlia ➲ http://kademlia.scs.cs.nyu.edun

Biztosítsuk be hálózatunkat a Kazaa ellen!

Ha a tűzfalak megkerüléséről van szó,
a Kazaa géptől-gépig rendszer igen ravasz tud lenni. Csakhogy nem eléggé.

Manapság a legnépszerűbb fájlmegosztó alkalmazás a Kazaa. Az ilyen típusú alkalmazásokat géptől-gépig (peer-to-peer, P2P) programnak nevezik, amelyek lehetővé teszik, hogy a felhasználók egymás gépén keressenek, és állományokat töltsenek le. A Kazaa programot jelenleg arra használják a legtöbbször, hogy hangállományokat terjesszenek a szerzői jogok megsértésével.
A Kazaa FastTrack néven ismert üzleti hálózati protokollját számos hasonló termék készítőjének is átadta (licencelte), például a iMesh és a Grokster programoknak. Elérhető továbbá a Kazaa „lebutított" változata is, KazaaLite néven. Rengeteg egyéb P2P-alkalmazás is létezik, de a FastTrack család messze a legnépszerűbb közülük, és a legnehezebben állítható meg olyan csomagszűrő tűzfalakkal, mint a Linux `iptables`.
Számos hálózati gazda szívesen letiltaná a tűzfalán a P2P-forgalmat a nagy sávszélesség-használat, az ellenőrzés nélküli fájlcseréből eredő biztonsági rések és a jogtulajdonosok esetleges jogi ellenlépései miatt. Csakhogy ez nem olyan egyszerű, mint amilyennek hangzik. Ha az interneten rákeresünk az `iptables` alapú FastTrack forgalomblokkolás témára, olyasféle válaszokat fogunk találni, mint „tiltsuk le a 1214 kaput", „készítsünk rendszabályt, és büntessük meg a gazembereket", vagy egyszerűen „ezt nem lehet megcsinálni". Bár a 1214-es kapu letiltása valóban működött a FastTrack korai változataival, a mai változatoknak már nem árt. Ennél kifinomultabb megoldásra van szükségünk. Néhány proxyalapú tűzfal képes ugyan megállítani a FastTrack forgalmát, de az `iptables` alapú tűzfalak esetében megoldást kell találnunk néhány nehézségre. Ebben a cikkben egy új, `P2Pwall` névre hallgató nyílt forrású projektet szeretnénk bemutatni. A projekt célja egy olyan program kifejlesztése, amely képes meggátolni a hálózatunkon próbálkozó P2P-ügyfeleknek a külső gépekhez való csatlakozását. A program `ftwall` összetevője meggátolja a FastTrack-forgalmat. További összetevők is készülnek, amelyekkel más P2P-protokollok ellen lehet védekezni. A fejlesztők között mindenkit szívesen látunk! A programot a következő FastTrack-ügyfelekkel próbáltuk ki: Kazaa 2.1.1, Kazaa 2.5, KazaaLite 2.0.2, iMesh 4.1 (build 132) és Grokster 1.7.

A tűzfalak nehezen boldogulnak a FastTrackkel

A korszerű Linux-terjesztések általában tartalmazzák a Netfilter és az `iptables`-eszközöket. Ezek az eszközök együtt lehetővé teszik, hogy Linux-rendszerünket egyszerű, de hatékony tűzfalként használjuk; csakhogy a FastTrack hálózati protokoll néhány érdekes kihívást rejt magában, amelyek a következők:

- nem használ állandó kapuszámot;
- a kommunikációja nem korlátozódik kis számú gépre: kétszáz gép címét tárolja és induláskor valamennyihez megpróbál csatlakozni; a lista időnként változik és minden gépen más és más;
- a gépkereső eljárás központi adattártól független;
- a protokoll kulcsfontosságú részei erős titkosítást alkalmaznak.

A tűzfalak általában két filozófia egyikét alkalmazzák. Az első igen szigorú: minden kaput letilt, kivéve néhány szükségeset. A második engedékenyebb és aszimmetrikus: szinte bármilyen kimenő kapcsolatot engedélyez, ugyanakkor csaknem minden bejövőt meggátol. Bármelyik megközelítésről legyen szó, a kapukat fürgén váltó FastTrack körbeszimatol, és kihasználja a legálisan nyitva lévő kapukat. Akár a 80-as kaput is használhatja. A szigorú szabályrendszerrel és egy 80-as kapun futtatott proxy együttes alkalmazásával ugyan megállíthatjuk a FastTracket, de ez a megközelítés már túlságosan kötött az olyan hálózatok számára, amelyek tartani szeretnék magukat az engedékeny elképzeléshez, mégis szívesen blokkolnák a P2P-forgalmat.

A P2Pwall projekt ftwall programja

A P2Pwall projekt ezeket a nehézségeket próbálja meg orvosolni azáltal, hogy számos, a P2P-forgalom szűrésével foglalkozó eszközt és dokumentációt nyújt. A FastTrack szűrő `ftwall` lenne az első ilyen eszköz, amelyet a GPL engedély oltalma alatt tölthetünk le a ➲ http://p2pwall.sourceforge.net címről. Az `ftwall` a `QUEUE` cél használatával kapcsolódik a `iptables`-hez. Megvizsgálja a tűzfalon keresztülküldött csomagokat, és a FastTrack protokoll jellemzőinek ismerete alapján eldönti, hogy továbbíthatók-e vagy el kell-e vetni őket. Egyúttal megpróbál a hálózatunkról kifele (így egyúttal befele) igyekvő minden FastTrack-forgalmat letiltani.
Az `ftwall` feladata a kifelé menő FastTrack-kapcsolatok tiltása, mivel feltételezi, hogy a bejövő kapcsolatokat az `iptables` szabályok már eleve meggátolják. Számos tűzfal használ általános blokkolást a befele érkező kapcsolatokra, ahol csak bizonyos számú kiszolgálókapcsolatot engedélyez. Csakhogy ha a belső FastTrack-ügyfél kezdeményez kapcsolatot kifele, a külső géphez, a kívülálló a már meglévő kapcsolaton keresztül visszahívhatja a belső gépet. Így aztán, ha a tűzfalunkra rábízhatjuk a bejövő kapcsolatok blokkolását, az `ftwall` pedig átnézi a kifelé menőket, a probléma megoldható; csak arra kell ügyelnünk, hogy mindkét elem a helyén legyen.
Az `ftwall` telepítése és beállítása annyiból áll, hogy letöltjük a forrást, lefordítjuk és megírunk néhány `iptables` szabályt. Esetleg nehézséget okozhat, hogy a rendszer egyik választható szolgáltatásának a használatához az `ip_string` modulnak benne kell lennie a rendszermagban. A modul jelenleg kísérleti szakaszban van, így sok Linux-terjesztés nem is tartalmazza. Ha használni szeretnénk, valószínűleg magunknak kell bíbelődnünk a felrakásával. További útmutatást a P2Pwall honlapján találunk.

Az iptables QUEUE (sor) célja

Amikor egy `iptables` szabály célként `QUEUE`-t ad meg, a szabállyal egyezést mutató minden csomag valamilyen alkalmazás (például az `ftwall`) által gyűjtött sorba kerül.
A program azután eldobhatja a csomagot vagy további ellenőrzésre és továbbításra adhatja vissza a Netfilternek. A szerke-

zetet meghívó jellemző szabály a következő alakú lesz:

```
iptables -A FORWARD -p tcp -i eth0
➥-dport 123 -syn -j QUEUE
```

Ha a fenti szabály érvényben van, az `eth0`-hoz csatlakozó hálózatról érkező és a távoli gépen a 123-as kaput célzó valamennyi SYN csomag először a programhoz kerül. A program első ízben beolvassa a csomagot, majd a `libipq` könyvtár és a `ip_queue` modul segítségével visszaküldi az ítéletét.
A `QUEUE` szabványos eleme az `iptables`-nek, amelyet a legtöbb terjesztésben megtalálhatunk. Ha ellenőrizni szeretnénk, hogy a rendszerünkön létezik-e ilyesmi, gépeljük be az `insmod ip_queue` parancsot, majd ellenőrizzük, hogy nem kaptunk-e hibaüzenetet. További részleteket olvashatunk a ➲ http://www.netfilter.org/documentation/FAQ/netfilter-faq-4.html címen elérhető Netfilter dokumentumban.

Az ftwall működése

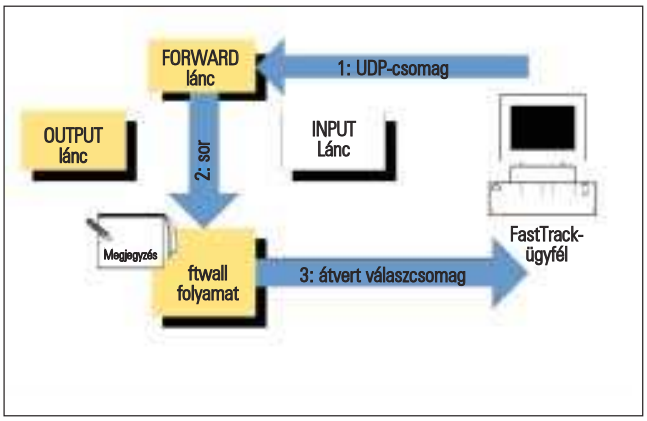
Ha ismertetni szeretnénk az `ftwall` szerkezetét, lépésről lépésre végig kell haladnunk a FastTrack-kapcsolat logikájának bizonyos részének az ismertetésén. A FastTrack három különböző megközelítést alkalmaz a gépekkel való kapcsolatteremtésre nézve: UDP-csomagok áradatát (flood), párhuzamos TCP/IP kapcsolatokat és a kicsit hagyományosabb TCP/IP kapcsolati mintát. Ha úgy érzi, hogy akadályozzák, a program egy másik módra vált át. Az `ftwall` az ügyfeleket igyekszik a lehető legtovább az első módban tartani, hiszen ezt a legkönnyebb azonosítani, valamint lehetővé teszi a célgépek listájának az elkészítését.
Az ügyfél induláskor nagyszámú UDP-csomagot küld keresztül a tűzfalon, amelyeket hosszuk és tartalmuk alapján beazonosíthatunk. A Netfilter ezeket az `ftwall`-on keresztüli feldolgozásba sorolja át (*1. ábra*). Ezután az `ftwall` belső feljegyzést készít a csomagok forrás- és célcímeiről, majd hamis választ küld az ügyfélnek. Ezáltal megakadályozza, hogy az ügyfél arra a következtetésre jusson, hogy az UDP-csomagokat tűzfal blokkolja, és így egy kicsit tovább fut az első módban.
Feltételezve, hogy a saját hálózati csatolófelületünk az `eth0`, az átsorolást a következő `iptables` szabállyal adhatjuk meg:

```
iptables -A FORWARD -p udp -i eth0 -j QUEUE
```

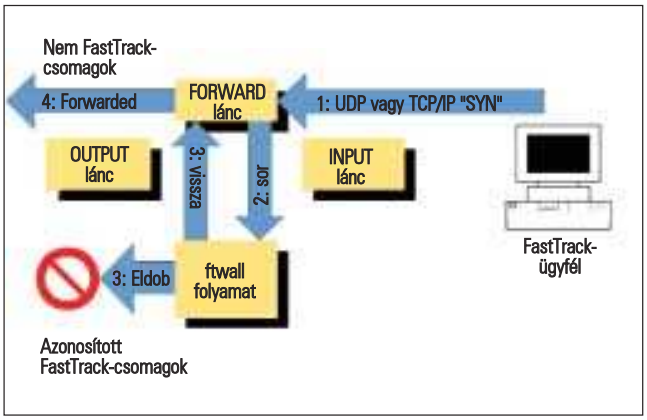
Miután a FastTrack megkapja a hamis választ, UDP-n keresztül megpróbál további adatokhoz jutni, majd ugyanehhez a címhez megkísérli létrehozni a TCP/IP-kapcsolatot. Ezek az UDP- és TCP-csomagok ugyancsak az `ftwall`-hoz kerülnek, amely most már tudja, hogy a célcímek a FastTrackre hivatkoznak, így eldobhatja őket. Az egyéb UDP-, nem FastTrack-csomagok és TCP/IP SYN-csomagok további ellenőrzésre visszakerülnek a Netfilterhez, s ezután céljuk felé továbbítódnak.
A következő szabály a SYN-eket átsorolja az `ftwall`-ba:

```
iptables -A FORWARD -p tcp -i eth0 --syn -j
QUEUE
```

Az ügyfél ezt az UDP és SYN sorozatot ismétli egy ideig – általában (de nem mindig) addig, míg a listájában található valamennyi ismert címet legalább egyszer ki nem próbálta. Ez egyben azt jelenti, hogy a címeket most már az `ftwall` is ismeri, és mint szűrendőket megjegyzi.
Idővel az ügyfél taktikát változtat és erős titkosítással felvértezett, párhuzamos TCP/IP-kapcsolatra vált át. Az `ftwall`

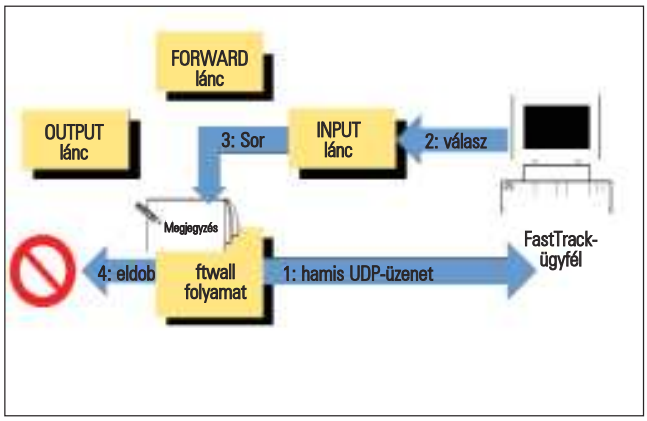


1. ábra Az UDP-csomagok nyitóképe



2. ábra Egyéb UDP- és TCP/IP SYN-csomagok

továbbra is gátol minden kapcsolatot az előző szakaszban feljegyzett címekre. Minden más cím esetében az egyetlen szóba jöhető bizonyíték, amely beazonosíthat egy FastTrack-kapcsolatot, az az, ha rövid idő alatt nagyszámú SYN-csomagot találunk valahol. Ha az `ftwall` kizárólag az UDP-csomagokra alapozná a blokkolást, vereséget szenvedne, különösen, ha az ügyfél az első szakaszban nem próbálta végig az összes ismert címet. A gond megoldása az időzár.
Ebben az új módban az ügyfél keverve próbál TCP-kapcsolatokat megnyitni az `ftwall` által ismert IP-címekre és a még ki nem derítettekre (amennyiben létezik ilyen). Az `ftwall` megjegyzi azt az időt, amikor a legutóbbi ismert címet megpróbálták elérni, és egy előre megadható ideig minden TCP/IP-kapcsolatot blokkol arról a forrás IP-címről. Minden SYN-csomag, ami ismert címre igyekezne, újra nullára állítja a számlálót. Amennyiben a kapcsolatot kellően sűrűn próbálják meg létrehozni, az `ftwall` folyamatosan blokkolni fog.
Ennek a módszernek a mellékhatásaként a lator gép minden TCP/IP-kapcsolata blokkolva lesz: mindaddig amíg a FastTrack próbál kapcsolódni, ideértve web- és az FTP-helyek elérését is. Mondhatjuk azonban, hogy ez elfogadható, hiszen a munkaállomás használója megszegi a szervezet szabályzatát. Amint az ügyfélalkalmazást lezárják, a számláló többé nem frissül, és a lejártát követően a TCP/IP-kapcsolatok ismét engedélyezettek lesznek. Az alapértelmezett beállítások szerint ez két percet jelent.
Miután a FastTrack egy ideig ebben a módban futott, arra a következtetésre fog jutni, hogy a párhuzamos stílusú kapcsolat problémákat okoz, ezért átvált a harmadik módba.



3. ábra UDP FastTrack-állapotpróba

Lelassítja a kapcsolatkereső próbálkozások sűrűségét, és a hagyományosabb, egyszerre csak egy címmel próbálkozó és a próbák között néhány másodperces szünetet tartó megoldást választja. Ez az új megközelítés minden eddigi elképzelésünket romba dönti, és az ügyfél időnként kijut. Ez akár egy órát is igénybe vehet, de azok az ügyfelek, amelyek nem árulják el idejekorán az összes címüket, ebben a szakaszban jó eséllyel képesek lesznek kapcsolatot kiépíteni. És ha már egyetlen egy kapcsolat kiépült, egy teljesen új címkészlet töltődik le. Ettől kezdve ott tartunk, mintha semmiféle blokkolást sem csináltunk volna.

A harmadik mód legyőzéséhez az `ftwall`-nak további adatokra van szüksége, amelyekből megállapíthatja, hogy a FastTrack még használatban van-e. Az egyik lehetséges módszer alkalmazásához még egy kis átejtésre van szükség. Az `ftwall` időről időre egy UDP-csomagot küld az ügyfélnek, mégpedig a pontos másolatát annak, amit az ügyfél maga használt a társához irányuló kapcsolat megnyitásához (*3. ábra*). Amennyiben a gépen fut a FastTrack program, egy könnyen felismerhető csomaggal válaszol, így az időzítő ismét alaphelyzetbe áll vissza. A viszonylag kisszámú és -méretű kutatócsomagok csak kevéssé terhelik a hálózatot.

Minthogy ezt a csomagot nem akarjuk nyilvános címre továbbítani, hanem magának a tűzfalnak szánjuk, létre kell hoznunk egy `iptables` szabályt az `INPUT` láncban, hogy az végül az `ftwall`-hoz kerüljön. A felhasználandó szabály:

```
iptables -A INPUT -p udp -i eth0 -j QUEUE
```

Így az ügyfelet folyamatosan hálózaton kívül tartjuk, de a megoldás nem igazán hatékony. Semmi mást nem kell tennünk, csak helyes időzárértéket alkalmaznunk, úgy, hogy az UDP-csomagokat akkor küldjük, amikor az idő körülbelül félig letelt, és az ügyfelet folyamatosan blokkolva tarthatjuk.

Kirakósjátékunk utolsó darabja egy biztonsági háló, amire elméletileg soha nincs szükség. A fenti logika felismerhető UDP-csomagokra alapoz, amelyek az `ftwall`-t a szükséges adatokkal látják el, de gondolnunk kell arra az esetre is, amikor ezek a csomagok egyáltalán nem érkeznek meg – például azért, mert a felhasználó az UDP-átvitelt kikapcsolja a munkaállomás tűzfalán. Ebben az esetben semmilyen módon nem tudhatjuk meg a társgépek címeit.

Ilyenkor is maradt még egy lehetőségünk: vizsgáljunk meg minden TCP/IP-adatcsomagot, és próbáljuk meg felfedezni bennük a fájlok átvitelére utaló nyomokat. A FastTrack titkosítási eljárása csak a kapcsolatteremtő kézfogásra és a keresésekre korlátozódik. A megosztott állományok egyszerű szöveges HTTP formátumban utaznak, de nincsenek a 80-as kapuhoz kötve. A HTTP-kérelemfejlécek számos mezőt tartalmaznak, amelyek a FastTrack-felhasználót, a protokollt és a szupercsomópont címét tartalmazzák (ez az a csomópont, amelyik az indexadatokat nyújtja). Amennyiben ezeket a csomagokat ellenőriztetjük, az `ftwall`-lal be fogja azonosítani közülük azokat, amelyek gyanúsan hasonlítanak FastTrack-fájlletöltés elejére. A HTTP-fejlécekben tárolt adatokból a blokkolandó címek listájára felveszi a cél és a szupercsomópont IP-címét, az ügyfelet pedig beszúrja azoknak a listájába, akikre az időzármódszert alkalmazni kell.

A telepítés áttekintése

A `ftwall` telepítésének folyamatát részletesen bemutatja a programhoz csatolt ***INSTALL*** állomány, illetve elolvashatjuk a projekt weblapján, de röviden a következő lépéseket kell megtenni:

- Töltsük le a forrásokat a ➲ http://P2Pwall.sourceforge.net címről, és csomagoljuk ki őket.
- Amennyiben még nincs telepítve, telepítsük a ***libipq*** könyvtárat. Néhány rendszeren (ide tartozik a Red Hat 7.x és 8) ez egyet jelent az `iptables` forrásának letöltésével és befordításával.
- Fordítsuk le és telepítsük az `ftwall`-t a `make` és `make install` parancsokkal.
- Adjunk egy bejegyzést a */etc/rc3.d* behúzófájlba, amely majd elindítja az `ftwall`-t.
- Ellenőrizzük, hogy elérhető-e a `QUEUE` mechanizmus, és tegyük fel, ha nem. A legtöbb mai Linuxban a helyén szokott lenni, de a rendszermag foltozásával és újrafordításával a többihez is hozzá lehet adni.
- Készítsük el az `INPUT` és `FORWARD` láncokba kerülő szabályokat.
- Ha „biztos, ami biztos” alapon telepíteni szeretnénk a fájlletöltések HTTP-fejlécének a vizsgálatát végző lehetőséget is, például abban az esetben, ha a hálózatunkon az UDP nem működne, adjuk a rendszermaghoz és az `iptables`-hez a `string` modult. Ehhez a rendszermagot meg kell foltoznunk és újra kell fordítanunk.
- Indítsuk újra a gépünket.

Összefoglalás

Az `ftwall` az írásunk születésekor használatos valamennyi FastTrack-ügyféllel megbirkózik. Elképzelhető, hogy a FastTrack-protokoll a jövőben megváltozik, ebben az esetben az `ftwall`-t is valószínűleg módosítani kell.

A megközelítés hátránya, hogy kizárólag a FastTrack-rendszerekre összpontosít; igaz, a P2Pwall projekt egyik célja, hogy a jövőben más P2P-protokollokra is kiterjessze a hatáskörét. Amennyiben valaki hajlandóságot érezne magában, és részt kívánna venni egy ilyen irányú fejlesztésben, írjon nekem levelet a chris@lowth.com címre.

Linux Journal 2003. október, 114. szám

Chris Lowth (chris@lowth.com)
az Intercai Mondiale (➲ http://www.intercai.co.uk) Egy angol telekommunikációs, IT- és üzleti tanácsadással foglalkozó cég munkatársa. Feleségével, három fiával és golden labradorjával Londonban él.

Bakelitről digitálisra

Egy jó bakelitlemez hangminőségét semmi nem múlhatja felül, ám ha utazunk, mégis kényelmesebb egy CD-t vagy egy Ogg Vorbis formátumú másolatot magunkkal vinni.

Tervezetem elindítása ***Dory Previn***-nek köszönhető. Az 1970-es években nagyszerű felvételeket készített, de az 1990-es években ezek CD-n sajnos már nem voltak elérhetők. Kicsit utánanéztem a dolognak, és rátaláltam ***Anne Bezemer*** és ***Ton Le*** Gramofile nevű csomagjára, amelyet kimondottan bakelitlemezek CD-re való átírásához készítettek. ***C. R. Johnson*** `xmcd2make` programja később a Gramofile szolgáltatásait bővítette ki. Tekintsük át az általa nyújtott lehetőségeket:

- Olcsó és tartós adathordozó használata zenék tárolására.
- Minden zeneszám egyedi elérésének biztosítása, időzítési adatokkal.
- Kódolás Ogg vagy MP3 formátumba.
- Recsegést csökkentő szűrés végzése egy-egy zeneszámra vagy a teljes albumra.
- A nem kívánt számok kihagyása, a zeneszámok sorrendjének megváltoztatása.
- Két album felvétele egy lemezre.

Az eszközök előkészítése

A hangminőséget több tényező is befolyásolja, ezekről a Linux Audio Quality HOWTO tartalmaz bővebb tájékoztatást, ám egy jó minőségű hangkártyára és kiváló támogatással rendelkező illesztőprogramokra biztosan szükség van. A kártyát a többi kártyától távol kell elhelyezni, így a lehető legkevesebb zajt szedi össze tőlük. Én erre a célra egy külön számítógépet használok, amiben csak két kártya van: az első PCI foglalatot a VGA-kártya foglalja el, az utolsóba pedig egy SoundBlaster Live! került. Ha olyan számítógépet akarunk a bakelitlemezek digitalizálására használni, amely más célokat is szolgál, akkor a 2.4-es sorozatba tartozó rendszermagot válasszunk. Tegyük fel a `preempt-kernel` és a `lock-break` foltokat (lásd a *Kapcsolódó címeket*), válasszunk megfelelő processzortípust, majd fordítsuk le és telepítsük a rendszermagot. A jó hallással rendelkezők számára talán nehezen hihető, de amikor az első bakelit digitalizálásával elkészültem, kellemes meglepetést okozott a CD hangminősége.

A szükséges programok

Ha csupán CD-t szeretnénk írni, akkor a Gramofile is elegendő – a megfelelő RPM vagy deb fájlból telepítsük. Az *1. képen* a főmenü látható; az egyes lépéseket később részletesebben is ismertetem. A Gramofile az `xmixer` nevű keverőt is keresi, amit az `mctools-lite` Debian-csomagban találtam meg, az RPM-világban pedig a Multimedia nevű csomagban lapul. Természetesen egy másik konzolon vagy ablakban futtatva bármilyen más keverőt is használhatunk.

A nem feltétlenül szükséges programok, amik jó, ha kéznél vannak

A zeneszámok Ogg vagy MP3 formátumba való átalakítása előtt fontoljuk meg az `xmcd2make` telepítését, valamint azoknak a programoknak a felpakolását, amelyektől függ. Ha az elején hajlandók vagyunk egy kicsit vacakolni a dolgokkal, később az egyes albumok feldolgozása sokkal egyszerűbb lesz. Az `xmcd2make` parancsfájlok például jó szolgálatot tehetnek, és a telepítésük is nagyjából a `make install` parancs kiadásából áll. Sajnos működni nem fognak, amíg a rendszerre fel nem kerül a Swig, az `oggenc` (Ogg-kódoláshoz) vagy a Lame (MP3-készítéshez), az `mpgtx` és a `Getopt::Long` Perl-modul. Az `xmcd2make` használatához a Gramofile különleges változatát kell telepíteni, amely ***perl-swig*** kiterjesztésekkel is rendelkezik – ezt a változat nevében egy P betű jelzi. Lássunk tehát munkához! Mivel a gépet csak egy célra használtam, a telepítéseket rendszergazdaként végeztem, a csomagokat a ***/usr/local*** könyvtárból bontottam ki.

A Swig telepítésével nem volt gond, de a Gramofile telepítésekor elakadtam. A legújabb, 1.3.17-es változattal próbálkoztam, de a `gramofile make perl-swig` parancs végrehajtása hibával állt le. Egy régebbi Debian-változatot, az 1.1.p883-4-est választva (az `apt-get install swig` paranccsal) a `make` sikeresen végrehajtódott. Mindezt kézzel az alábbi parancsokkal végezhetjük el:

```
tar xvzf swig1.1-883.tar.gz
cd  SWIG1.1-883
./configure
make
make install
```

Az `oggenc` RPM vagy deb csomagok formájában érhető el – igaz, a nevek megtévesztők lehetnek. Debian alatt az `apt-get install vorbis-tools libvorbis0` parancsot kell kiadni. A Lame felkutatása szabadalmi okok miatt nehézkesebb; az Ogg minden szempontból tiszta és jó választás, így inkább ennek a használatát javaslom.

Az `mpgtx` egy parancssori MPEG-eszközkészlet, amelyet Debian alatt az `apt-get install mpgtx` paranccsal telepíthetünk. Én az 1.3-as változat mellett döntöttem, amelyet viszont forrásból, a Swighez hasonlóan a klasszikus

```
tar
./configure
make
make install
```

parancsokkal telepítettem. A művelet közben ugyan több oldalnyi figyelmeztetés jelent meg, de a telepítés sikeresen befejeződött.

A `Getopt::Long` Perl-modul a Debian 5.6.1 csomag része, és remélhetőleg más rendszereken is megtalálható, az én gépemen a ***/usr/share/perl/5.6.1/Getopt/Long.pm*** úton érhető el.

A ***perl-swig*** kiterjesztésekkel ellátott Gramofile kézi telepítését csak bátraknak ajánlom. Szükséges hozzá az `ncurses5-dev`,

és sajnos nem mászik fel magától a gépre. Mivel ismernünk kell a Perl CORE helyét, a következőkkel próbálkozzunk:

```
cd /usr/lib
find -name CORE
./perl/5.6.1/CORE
```

A fentiek szerint az én gépemen a Perl CORE a */usr/lib/perl/5.6.1/CORE* könyvtárban található. A Gramofile *.tar* állományának kibontása után módosítani kell a *perl-swig* alkönyvtárban található `Makefile` állományt, a `PERLCORE = -I/usr/...` sort olyan módon megváltoztatva, hogy a saját gépünkön lévő telepítésre mutasson:

```
tar xvzf gramofile-1.6P
cd gramofile-1.6P
cd perl-swig
(a Makefile szerkesztése)
cd ..
make
make perl-swig
```

Következzen a futtatható állománynak egy olyan könyvtárba történő átmásolása, amely szerepel a `$PATH` környezeti változóban:

```
cp gramofile bplay_gramo brec_gramo /usr/bin
```

Most lépjünk át a *perl-swig* alkönyvtárba, majd a benne található két fájlt másoljuk át a Perl elérési útban megadott könyvtárba. Elsőként természetesen a Perl elérési utat kell meghatározni:

```
perl -e ´print join("\n",@INC), "\n"´
```

Saját debianos gépemen így nézett ki a dolog:

```
cd perl-swig
mkdir /usr/local/lib/site_perl
cp Gramofile.pm Gramofile.so
/usr/local/lib/site_perl
```

Végül az `xmcd2make` következik:

```
tar xvzf xmcd2make-0.4.tar.gz
cd xmcd2make-0.4
make install
```

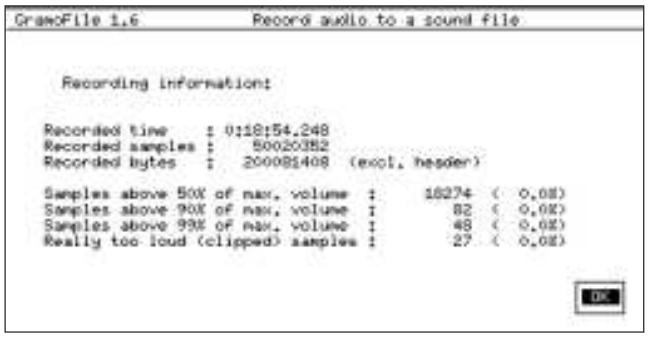
Az `xmcd2make` alapesetben 128-as bitszámmal dolgozik, ám én – elfogadva, hogy nagyobb fájlokat kapok és a kódolás is tovább tart – nagyobbat szeretek használni, ezért a */usr/local/bin/xmcd2make* fájlban a bitszámot átírtam 224-re:

```
# $bitrate = 128;
$bitrate = 224;
```

Ha belefér még egy program, javaslom a `umix` nevű keverő telepítését, mivel ennek konzolos változata is van, a szinteket finoman és megismételhető módon lehet állítani benne, és képes rá, hogy egyetlen gombnyomásra az összes beállítást mentse vagy visszatöltse. Így a bakelitlemez CD-re írásának teljes folyamata egy olcsóbb, X nélküli számítógépen is lejátszódhat. A keverő alapértelmezett elérési útja a */dev/sound/mixer*, ezt a következő módon kell majd módosítani:

```
./configure --with-mixer-dev=/dev/mixer
```

1. kép A Gramofile főmenüje



2. kép A Gramofile rögzítési adatai

Ha normál felhasználóként szeretnénk beállításokat betölteni vagy menteni, akkor a `umixet` a beállításfájl nevének megadásával kell indítanunk, például `umix -f $HOME/umixrc` Az S gombbal a pillanatnyi beállításokat lehet menteni, az L gombbal pedig az utoljára mentetteket lehet visszatölteni.

A felvétel menete

A következőkben Dory On My Way To Where című albumának példájával szemléltetem, hogyan kerülhet egy-egy album először a merevlemezre, hogyan lesz belőle *wav*-fájlok kisebb gyűjteménye, amelyekből elkészül egyrészt a zenei CD, másrészt a számítógépeken és hordozható lejátszókkal jobban kezelhető Ogg-fájlok. Az `xmcd2make` számára a „where" alapnevet adtam meg, ebből származtathatók a fájlok nevei is.

- A számítógépet helyezzük a lemezjátszó közelébe, majd a kimenetét egy jó minőségű, árnyékolt kábellel kössük a hangkártya bemenetéhez. Az összeköttetés létrehozásához nálam kettős RCA-sztereo minicsatlakozó-átalakítóra volt szükség.
- Xterm alatt vagy konzolról töltsük be a keverőprogramot. Egy másik konzolon vagy ablakban lépjünk át egy olyan könyvtárba, amely egy jó sok szabad helyet tartalmazó lemezrészen van, majd töltsük be a Gramofile-t.
- A keverőt állítsuk „line in" (bemenet) felvételi módba, a többi csatornát pedig némítsuk el. A bemeneti erősítést (igain) szintén érdemes növelni, így csökkenthető a háttérzaj.
- A Gramofile ***Record audio*** (hangfelvétel) módjában készítsünk próbafelvételt, eközben a keverő szintjeit a Gramofile szintmérői alapján emeljük közel a legnagyobbra.
- Állítsuk le a felvételt, ellenőrizzük, hogy a minták kellően nagy része a legnagyobb hangerő fele és kis része a 90 százaléka fölé esett-e, illetve azt, hogy nincs-e 99 százalék fölötti minta (*2. kép*).
- Valamilyen beszédes név megadásával – a példában ez a

where1.wav – állítsuk be a Gramofile-t a lemez első oldalának felvételére, indítsuk el a lejátszást, majd a felvételt is.

- Amikor lejárt az első oldal, állítsuk le a Gramofile-t, és ellenőrizzük a minták minőségét. Ha nem sikerült megfelelőre, állítsunk a hangerőszinteken. A hangerő pillanatnyi nagy ugrásait a lemez karcolásai, pattogásai okozzák, ha nincs túl sok ilyen hiba, akkor a felvétel elfogadható.
- Vegyük fel a második oldalt is, esetemben ez a *where2.wav* fájlba került.

Most két, egyenként körülbelül 200 MB méretű állománnyal rendelkezünk, amelyek a lemez két oldalának digitális másolatát tartalmazzák. Ideje megvizsgálni, vajon a lemez lejátszása mekkora zajjal járt, illetve a bakelitre annyira jellemző pattogások mennyire uralják az összképet. Ha az egész album zajos, akkor a hanganyagot legalább a Gramofile szűrőinek egyikén keresztül kell zavarnunk. Ha csak bizonyos zeneszámok pattognak – általában mindkét oldal első száma szokott zajos lenni –, akkor a zajszűréssel várjuk meg a hanganyag számokra osztását. A Gramofile leírásában (Signproc.txt) elképesztően jól sikerült az összes szűrő működéséről és annak elméleti hátteréről szóló rész. A ***Process the audio signa***l (jelfeldolgozás) parancs választásakor észre fogjuk venni, hogy a ***Conditional Median Filter II*** szűrő már ki van jelölve. Ez az összes közül a legkifinomultabb, jómagam kiváló eredményeket értem el vele. A pattogások ugyan nem tűntek el teljesen, de jelentősen gyengültek. Több szűrőt is lehet használni, illetve ugyanaz a szűrő kétszeresen is használható. Én inkább az egyszeres alkalmazást javaslom, mert amikor a ***Conditional Median Filter II*** szűrőt kétszeresen használtam, a zene minősége érezhetően romlott. Szerencsére az egész folyamat csak néhány percet igényel, így mindenki bátran kísérletezhet. Az eredeti fájl minden esetben megmarad, a szűrt állománynak pedig valamilyen beszédes nevet kell adni. Hallgassuk meg az új *wav*-fájlt. Ha tetszik az eredmény, töröljük le az eredetit, majd a szűrt fájlnak adjuk az eredeti fájl nevét. A számok kódolása előtt nyilván meg szeretnénk adni az előadó nevét, az album és az egyes számok címét. Ezek begépelésétől a ***freedb.org*** segítségével kímélhetjük meg magunkat. Ha az oldalon megtalálhatók az album adatai, kattintsunk az első szám címe felett található azonosító hivatkozásra, majd a felbukkanó oldalt szöveges fájlként mentsük. Természetesen a listát egyszerűen ki is másolhatjuk erről az `xmcd` oldalról, majd beilleszthetjük valamilyen szövegszerkesztőbe. A # (kettős kereszt) karakterrel kezdődő sorokat ilyenkor hagyjuk el. A címeket megfelelő névvel ellátott, egyszerű szövegfájlba kell menteni, abba a könyvtárba, amelyben a *wav*-fájlok is találhatók. Maradva a példánál, a fájl neve ***where.xmcd*** lesz. Bármelyik megoldás mellett döntünk is, csak a `DTITLE` és a `TTITLE` sorokra lesz szükségünk. Az ismeretlen lemezek címeinek beírásához én egy `xmcd` fájlt készítettem, amelyben csak `DTITLE=` és `TTITLE0=` – `TTITLE10=` sorok találhatók. Néhány perc elég hozzá, hogy készítsek belőle egy, a pillanatnyi album címével egyező nevű másolatot, majd a lemez borítójáról begépeljem a címeket. Ügyeljünk arra, hogy a zeneszámok számozása nullával kezdődik. Lássunk egy példát:

```
DTITLE=Dory Previn / On My Way To Where
TTITLE0=Scared To Be Alone
TTITLE1=I Ain´t His Child
```

Következő feladatunk az egyes oldalak zeneszámokra vágása. Aki egyszerű megoldást szeretne, az válassza a Gramofile ***Locate tracks*** (számok keresése) parancsát, válassza ki az első oldalt (***where1.wav***), kattintson a ***Next*** (tovább), majd a ***Start computation*** (feldolgozás indítása) gombra, és várjon türelemmel, amíg a program végig nem szalad a számokon. Ha a kapott szám nem egyezik meg a zeneszámok számával, akkor a feldolgozás elindítása előtt próbáljuk meg módosítani a beállításokat, például a zeneszámok közötti szünetet vegyük le 12 másodpercre vagy kevesebbre. A műveletet ismételjük meg a második oldallal is. Ha a kapott szám megfelel, válasszuk a ***Process the audio signal*** (hangadatok feldolgozása) parancsot, újra válasszuk ki az első oldal *wav*-fájlját, kattintsunk a ***Next*** gombra, majd ha az alapértelmezett szűrőt akarjuk használni a pattogások eltüntetésére, lépjünk át a ***Next*** (következő) lapra. Ha semmilyen szűrőt nem szeretnénk használni, akkor egy pillanatra álljunk meg, és az ***Available filters*** (elérhető szűrők) listából válasszuk a ***Copy only*** (csak másolás) pontot, majd nyomjuk le az ENTER billentyűt. Lépjünk át a ***Selected filters*** (kiválasztott szűrők) listára, jelöljük ki a ***Conditional Median Filter II*** szűrőt, majd az ***R*** gombbal távolítsuk el. Másfajta vagy további szűrőt hasonló módon választhatunk ki. A fájlok feldolgozásának elindításához a szűrők vagy a ***Copy only*** pont valamelyikét mindenképpen ki kell választanunk. Ha a ***Selected filters*** listában valamelyik szűrő ki van jelölve, akkor az ENTER lenyomásával módunk nyílik a szűrő beállításainak a megváltoztatására. Lépjünk át a ***Start*** lapra, majd nyomjuk le az ENTER-t. A folyamat végeztével minden számhoz egy-egy *wav*-fájlnak kell tartoznia, ezeket próbaképpen érdemes meghallgatni, majd indulhat a CD-re írás.

Ha az `xmcd2make` mellett döntöttünk, akkor a Gramofile-t be is zárhatjuk, ugyanis az `xmcd2make findtracks` parancsfájlja a `gramofile findtracks` szolgáltatásának a burkolója. A parancssorból futtassuk le a `findtracks where1.wav` parancsot, ezzel végigpásztázzuk az album első oldalát. A kimenetet hasonlítsuk össze a hivatalos zeneszámlistával. Írjuk be a `less *.tracks` parancsot, ekkor egy szöveges fájlt fogunk látni, amely az egyes számok kezdetének és végének az időpontját tartalmazza. Ha két szám véletlenül egybenmaradt, lépjünk vissza a parancssorba, majd egy vagy több kapcsoló értékét módosítva próbálkozzunk újra, például:

```
findtracks where1.wav --min-silence-blocks 12
```

Természetesen kézzel is szét lehet vágni a dalokat, illetve – lusták előnyben – az egész lemezoldalt egyetlen számként is fel

KAPCSOLÓDÓ CÍMEK

Gramofile kiterjesztésekkel vagy nélkülük
➲ http://panic.et.tudelft.nl/~costar/gramofile
CD-írás ➲ http://www.tldp.org/HOWTO/CD-Writing-HOWTO.html
Gramofile *perl-swig* kiterjesztéssel és az xmcd2make programmal ➲ ftp.freeengineer.org/pub/xmcd2make
Linux hangminőségi útmutató
➲ http://www.linuxdj.com/audio/quality
Rendszermagfoltok ➲ http://www.tech9.net/rml/linux
mpgtx ➲ http://mpgtx.sourceforge.net
Perl-modul telepítése
➲ http://www.perldoc.com/perl5.8.0/lib/CPAN.html
Swig ➲ http://www.swig.org
umix ➲ http://umix.sourceforge.net

lehet írni a CD-re. Ebben az esetben is a Gramofile szűrőjét kell használni, az előbbihez hasonlóan a ***Copy only*** pontot kell választani, ám a ***Split tracks*** (számokra osztás) beállítást le kell tiltani. Ekkor szűrés nem történik, ám a *wav*-fájlba bekerülnek azok az időzítési adatok, amelyek alapján a CD-író tudni fogja a tényleges játékidőt. Ha valamelyik számot a program kétfelé vágta, ezek összeillesztése gond nélkül megoldható. Az egyesítés után át kell számozni a többi számot, valamint a `Number_of_tracks=` sort is megfelelően módosítani kell. Az egész eljárást a második oldallal is meg kell ismételni. Ezt követően készíthetünk egy `Makefile` állományt, amellyel önműködően lefuttathatjuk a folyamat hátralévő részeit:

```
xmcd2make --basename where --counts 5,5
➥> Makefile
```

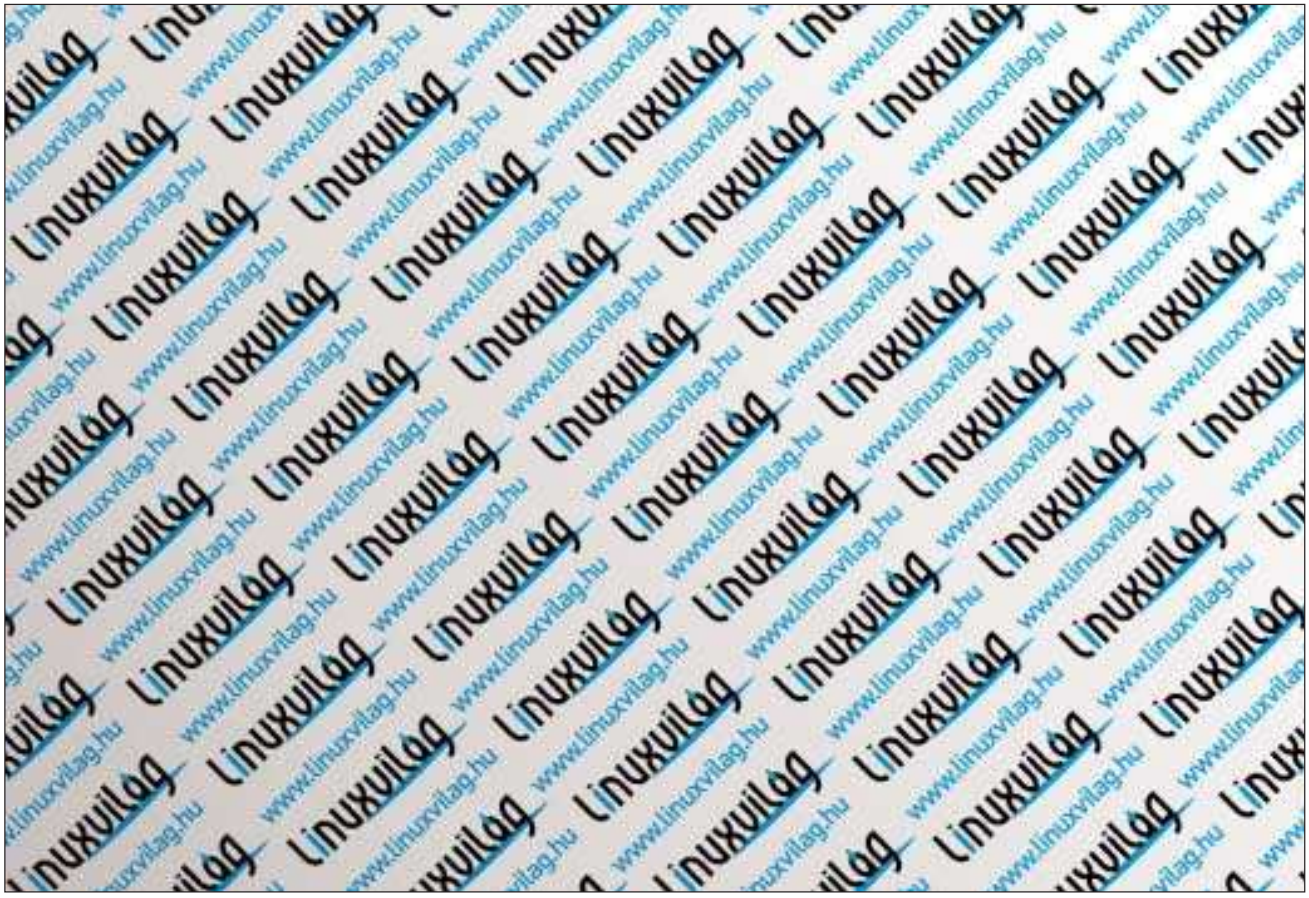
A paranccsal egy `Makefile` állományt hozunk létre, amely tartalmazza a számok és az album felosztásához szükséges időadatokat, az előadó nevét és a számok címeit; az utóbbiakat Ogg vagy MP3 készítéséhez is használhatjuk. A számlálók értékeinek az egyes zeneszámokat tartalmazó fájlokba írtakkal meg kell egyezniük. A bitszámot például a `--rate 192` kapcsolóval módosíthatjuk. Most mindössze a `make` parancsot kell kiadnunk, a program minden számot külön *wav*-fájlba másol, majd ezeket valamilyen beszédes névvel ellátva Ogg-fájlba kódolja. Lássuk példánk rövidített könyvtárlistáját:

```
Makefile                    9.5k
where1.wav                  196M
where1.wav.tracks           1.2k
where2.wav                  191M
where2.wav.tracks           1.2k
where_processed_101.wav     52M
101_Scared_To_Be_Alone.ogg  8.2M
```

Ha a számokra vágás után Ogg- helyett MP3-fájlokat szeretnénk kapni, akkor a `make mp3` parancsot kell kiadnunk.
A `make proc` parancs hatására csak a számokra vágás történik meg, amelyet követően a szükséges szűrőt egy-egy zeneszámra alkalmazhatjuk. Az eredeti, szűretlen fájl törlése és a szűrt átnevezése után a `make` paranccsal készíthetünk Ogg-fájlt a *wav*-állományból. A további kapcsolókkal az `xmcd2make --help` parancs segítségével ismerkedhetünk meg.
Nos, végeztünk is. Most már vannak *wav*-fájljaink, amelyeket CD-re tudunk írni, valamint Ogg-állományaink, melyeket megfelelő eszközzel le tudunk játszani vagy adat-CD-re tudunk írni. Ha a rendszert sikerült összeállítani, a további lemezek feldolgozása már rutinmunka. Egyébként 2002-ben Dory utolsó albuma végre megjelent CD-n is. Igaz, ez engem már nem érint, mert nekem van saját, Linux alatt készült lemezem.

Linux Journal 2003. szeptember, 113. szám

Tom Younker (tom@darecomputer.com)
A Georgia állambeli Atlantában él, felesége Mac-rajongó, a pincéje pedig linuxos gépekkel van tele. Tanácsadóként dolgozik.



Bemutatkozik a SHOUTcast

Saját internetes rádióállomást mindenkinek!

Sorozatunk előző részében igyekeztem felvázolni az internetes műsorszórás alapjait. Mit érne azonban önmagában a száraz elmélet, ha nem kóstolhatnánk bele a gyakorlat zamatos világába? Éppen ezért jelen cikkünkben a dolog közepébe vágunk, s a Nullsoft SHOUTcast nevű programjának a segítségével felépítünk egy, az internetről vagy a helyi hálózatról elérhető rádióállomást – hiszen az előző számban megjelent témaleírásnak köszönhetően minden szükséges adatot ismerünk az internetes műsorszórással kapcsolatban.

Azért választottam egy MP3 alapú műsorszórás bemutatását, mert ennek a feltételei állnak legközelebb a széles körű megvalósíthatósághoz, ami a gyakorlatban annyit teszt, hogy bárki kipróbálhatja, akinek a gépe rendelkezik valamilyen egyszerű hálózati csatlakozással. Ilyenkor csak annyi a dolgunk, hogy telepítjük a gépünkre a rádióállomást, majd egy másik masinával, amely TCP/IP-protokollon keresztül képes látni a miénket, hallgatóságként csatlakozhatunk az állomáshoz. Megjegyzem, a saját gépünkről is csatlakozhatunk a gépünkön futó rádióállomás-szolgáltatáshoz, de ugye az élmény kedvéért mégiscsak szebb lenne, ha fizikailag is egy másik ügyfél hallgatna bennünket. Ha valamilyen széles sávú kapcsolattal bírunk (ADSL, kábeltévés kapcsolat), nyugodtan szóljunk a szomszédnak, hogy próbaképpen csatlakozzon a gépünkhöz. A feltöltésre engedélyezett 16 kb elég ahhoz, hogy egy, legfeljebb két ember bárhonnan az internetről csatlakozzon a gépünkhöz. A modemes kapcsolat sajnos nem elég, de igazából egyáltalán nincs szükségünk internet-elérésre; megteszi, ha van egy helyi (LAN) kapcsolatunk a szobában, a lakásban, egy kollégiumban, netán össze vagyunk kötve a mellettünk lévő lakással, s él a TCP/IP-kapcsolat.

Mi is a SHOUTcast?

A program egy ingyenes adatfolyam-előállító és kiszolgáló, amit a Winampot is fejlesztő Nullsoft készített – alapvetően a windowsos Winamphoz, ám a program linuxos változata is elérhető. A program valójában két részből áll: az egyik a SHOUTcast kiszolgáló, amely a számára előállított adatfolyamot a távolról kapcsolódó gépek számára elérhetővé teszi, a másik része pedig egy olyan adatfolyam-készítő program (ez a mi esetünkben a TRANScast), amely kifejezetten a kiszolgálónak szolgáltatja az adatokat, ez pedig szétszórja azokat az interneten. Mindkét program elengedhetetlenül szükséges a működéshez, de nem muszáj egyazon gépről futniuk – csupán annyi kell, hogy a hálózaton keresztül lássák egymást. A SHOUTcast kiszolgáló egyébként nemcsak Windows vagy Linux, de FreeBSD, MacOS és Solaris felületeken is futtatható, a hozzá kapcsolódó adatfolyam-szolgáltató segédprogram tudása azonban különbözik az egyes rendszereken.

Telepítsük a programot!

A programok megtalálhatók a magazin 54. CD-mellékletének Magazin/SHUOTcast könyvtárában. A ***shoutcast-1-9-2-linux-glibc6.tar.gz*** nevű állomány tartalmazza a SHOUTcast kiszolgáló, az ***sc_trans_posix_040.tgz*** pedig a hozzá tartozó TRANScast nevű adatfolyam-szolgáltató programot. Egyébként mindkettő a ➲ http://www.shoutcast.com címről tölthető le, az újabb változatok keresése során is itt járhatunk sikerrel. Csomagoljuk ki az állományokat bármilyen tetszőleges könyvtárba (külön a két állomány tartalmát). A SHOUTcast kiszlgáló egy binárisból és egy beállításállományból áll, hasonlóképpen, mint a TRANScast, de a TRANScast rögtön három bináris állományt is magában foglal, amelyek közül mi az `sc_trans_linux` nevűt fogjuk majd használni.

A kicsomagolás után – ami egyben a telepítés – először testre kell szabnunk a kiszolgálót. Kezdjük is el! Nyissuk meg a kicsomagolt könyvtárban található ***sc_serv.conf*** fájlt, ez sok-sok megjegyzéssel tűzdelve tartalmazza a kiszolgáló beállításait. A seregnyi beállítási lehetőségre való tekintettel előbb csak azokat vesszük sorra, amelyekkel kapcsolatban feltétlenül vagy nagyon

A SHOUTcast házatája

ajánlott, hogy a működtetéshez testreszabjuk őket. Az egyes kényelmi szolgáltatásokra és a különleges szolgáltatások beállításának mikéntjére sorozatunk következő részében fogunk kitérni.

A SHOUTcast kiszolgáló alapvető beállításai

- `MaxUser`: az elejéről haladva az első beállítás a `MaxUser` nevű lehetőség, amely azt mondja meg, hogy legfeljebb hány hallgató csatlakozhat az állomásunkhoz. Ezt célszerű a rendelkezésre álló sávszélesség és a később sugározni kívánt adatfolyam méretének a függvényében beállítani. Számoljuk ki, hogy hányszor fér bele egy adott sávba egy adott bitráta, és aszerint módosítsuk. Minden futó kiszolgálópéldány pontosan egy bemeneti adatfolyamot fogad, és annyi kimenőt engedélyez, amennyit itt beállítunk.
- `Password`: itt tároljuk a kiszolgáló jelszavát. Csak olyan szolgáltatóprogram csatlakozhat a kiszolgálónkhoz, amelyik ismeri ezt a jelszót. Később látni fogjuk, hogy a TRANScast megfelelő helyén majd szintén be kell állítani a csatlakozási jelszót! Ezzel védekezhetünk az ellen, hogy illetéktelen személy hozzákapcsolja az általa előállított adatfolyamot a mi kiszolgálónkhoz.

Miért késik a hang?

Annak érdekében, hogy a pillanatnyi hálózati kimaradások vagy ideiglenes sávszélesség-csökkenések miatt a hallgatott adásban ne legyenek kiesések (szaggatás), a lejátszóprogramok a bejövő adatokat átmenetitárazzák, így ha kisebb fennakadás keletkezik az adattovábbításban (s mint tudjuk, ez a mindennapokban elég gyakori), addig az átmeneti tárolóban lévő hanganyagot játssza le, s amint újra helyreáll a kapcsolat, újra feltölti az átmeneti tárat. Ennek eredményeképpen mi, hallgatók mit sem veszünk észre belőle. Ehhez jön még az is, hogy a rádióállomás is alkalmaz némi átmeneti tárazást. Kivételes esetekben további késéseket okozhat, hogy továbbító (relay) kiszolgálók vannak a láncba iktatva, esetleg újra is tömörítik az adatfolyamot, s ez azt eredményezi, hogy nem mindegyik hallgató hallja szinkronban az adást. Mindennek a késleltetésnek nálunk akkor jelentkezik a hatása, ha a saját kiszolgálónkra csatlakozva tanulmányozzuk annak viselkedését: valamilyen változtatást végzünk az adásban, s figyeljük, hogy ez mit eredményez a hangszórókon. Ebben az esetben ne lepődjünk meg, ha a változtatás (új szám betöltése) csak 30–60 másodperc múlva történik meg. Ennyi kell ugyanis ahhoz, hogy az előre betárazott adatfolyam az átmeneti tárban kicserélődjék, s a mi fülünkhöz is már az új adás jusson el.

Ez a mező nem is maradhat üresen, kötelező kitölteni.

- `PortBase`: ez határozza meg, hogy melyik kaput használja a kiszolgáló. Erre a kapura csatlakozik mind a bemenő adatfolyam (amelyet a TRANScast hoz létre), mind a hallgatók. Fontos, hogy a beállított kapu és a tőle eggyel nagyobb sorszámú is szabad legyen, tehát egyetlen más program se használja az adott gépen. Ha egy gépen esetleg több rádióállomást is szeretnénk üzemeltetni, különböző kapuszámmal kell ellátnunk őket, s a csatlakozó hallgatóság e kapuszám alapján választhatja meg, hogy melyik szolgáltatást veszi igénybe. Például ilyen több példányban futtatott kiszolgálókkal tehetjük meg azt, hogy egyszerre sugárzunk egy jobb és egy gyengébb minőségű adatfolyamot. Akinek lassabb a kapcsolata, az a gyengébb adatfolyamot választja, így azok sem esnek el a szolgáltatástól, akik jó sávszélességgel rendelkeznek, s fontos számukra a minőség. (Az alapértelmezett kapuhoz nem kell hozzányúlnunk, nekünk megfelel ez a 8000-es kapu.)

Ezennel a fő beállítások végére is értünk, ezek után már csak el kell indítanunk az ott található binárist, és hagyni, hadd fusson – amennyiben nincs bemeneti adatfolyamunk, úgysem foglal le erőforrásokat.
(Megjegyzés: nem rendszergazdaként indítva a program futása rendszeresen megszakad, ha egy adatfolyamforrás csatlakozni szeretne hozzá.) Most azonban nézzük adatfolyam-szolgáltató programunkat, a TRANScastot!

A TRANScast

Ez a program arra szolgál, hogy a számára megadott forrásanyagot MP3 formátumúvá alakítsa, majd folyamatosan átküldje a már futó SHOUTcast kiszolgálónak. Ez az alkalmazás készíti el magát a rádióadást. A program alapvetően kétféleképpen működik: az egyik működési módban az általunk összeállított lejátszási lista tartalmát olvassa be, és adja tovább a kiszolgálónak, a másikban egy másik SHOUTcast-kiszolgálótól veszi át az adatot, kisebb méretűre (rosszabb minőségűre) tömöríti át, majd továbbadja az általunk futtatott SHOUTcast kiszolgálónak, így azok is élvezhetik az adást, akik lassúbb interneteléréssel csatlakoznak a gépünkhöz. Ha még mélyebbre ásunk, akkor láthatjuk, hogy a lejátszási listából történő adattovábbításnak is kétféle módja létezik: az egyik során MP3-fájlokat olvasunk be, s azt adjuk tovább, a másik mód csak linuxos rendszereken érhető el, ennek során az alkalmazás a hangkártya vonalbemenetére csatlakoztatott adatfolyamot alakítja át MP3-má. Ez azért jó, mert így például „élő” adások sugárzására is lehetőségünk nyílik, ugyanis a hangkártyára egy kisebb „stúdiót” köthetünk. A stúdió egy keverőpultból állhat, amelynek bemeneteire mikrofont, CD-lejátszót vagy egy másik számítógépet köthetünk – ezzel készíthetjük el az adást, s az így előállt műsorfolyamot kapja meg a kiszolgálógép.
Maga a program három részből áll: a linuxos binárisból, a hozzá tartozó beállításfájlból, végül a lejátszási listafájlokból, amelyek vagy a hangkártya bemenetét, vagy az MP3-as fájlok elérési útvonalait tartalmazzák, egymás alatt felsorolva.

A lejátszási lista beállításai

Mindenekelőtt kezdjük ezzel! Javaslom, tegyünk pár MP3-at a háttértárolóra, majd hozzunk létre egy lejátszási listát. A lista neve tetszőleges lehet, ugyanis később a TRANScast beállításai között mondjuk majd meg, hogy melyik állományt használja lejátszási lista gyanánt. A fájl formátuma egyértelműen kiderül az ***example.lst*** állományban található megjegyzésekből is. Egymás alatt soroljuk fel őket a teljes elérési úttal együtt. Hogy egy album esetén ne mindent egyenként kelljen felvinni, könnyítésképpen egy könyvtárlistát is beleirányíthatunk.
Abban az esetben, ha a hangkártya vonalbemenetét szeretnénk használni, csak egyetlen sort írjunk az MP3-fájlnevek helyére: ***DSP:/dev/audio***.
(Ha a hangkártya másképp lett beállítva, természetesen a vonalbemenet-eszköz kezelőállományát kell megadni.)
Figyelni kell arra, hogy a program a sorrendben legelső MP3-at valamiért kihagyja, tehát első körben egyszerűen nem játssza le.

A TRANScast beállítási lehetőségei

A program beállításai az ***sc_trans.conf*** fájlban találhatók. A változtatások elvégzéséhez nyissuk meg ezt a fájlt, és nézzük, mit is állíthatunk be:

- `PlaylistFile`: itt adhatjuk meg, hogy a program melyik fájlt kezelje lejátszási listaként. Kötelező megadni, hacsak nem továbbító üzemmódban használjuk. Adjuk meg annak a fájlnak a nevét, amelyet létrehoztunk (ha az ***example.lst***-t módosítottuk, ezt az értéket hagyjuk változatlanul).
- `ServerIP`, `ServerPort`: a SHOUTcast kiszolgálót futtató gép IP-címe és kapuszáma (lásd ***A SHOUTcast kiszolgáló alapvető beállításai*** című részben). A mi esetünkben a két program ugyanazon a gépen fut, így a ServerIP értékét állítsuk `localhost`-ra, a kapuszámot pedig arra az értékre, amelyet a kiszolgálónál beállítottunk. (Ha ott nem nyúltunk hozzá, akkor itt se tegyük!)
- `Password`: a jelszó a SHOUTcast kiszolgálóhoz, amely az adatfolya-

Jelek

A jelek (signals) valójában a folyamatok közötti kapcsolattartás szabványos eszközei, amelyek a legtöbb operációs rendszerben általánosan elfogadott üzenetküldési lehetőséget nyújtanak. Nincs ez másként a Linux, illetve Unix-rendszerekben sem; ezekben a `kill` parancs segítségével küldhetünk üzeneteket az egyes folyamatoknak, amelyek a rendszermagon keresztül jutnak el az adott folyamathoz. Eközben a rendszermag ellenőrzi a jel jogosultságát az adott folyamathoz, vagy önmaga avatkozik be. Erről a parancsról már biztosan sokan hallottak, a legtöbben arra használták, amire a neve alapján elsőként gondolunk: a folyamatok „megölésére", vagyis futásuk megszakítására. Ekkor ugyanis egy olyan jelet küldünk a folyamatnak, amelynek a tartalma a következő: fejezd be a futásodat! (Megjegyzem, hogy ez igazából egy különleges jel, amelyet a rendszermag maga értelmez és hajt végre.) Használatának módja a következő:

```
kill -s <üzenet> <folyamat azonosítója>
```

munkat elérhetővé teszi a kapcsolódó hallgatók számára. Azt a jelszót adjuk meg, amit a SHOUTcast kiszolgáló beállításánál már korábban szerepeltettünk. Figyeljünk oda a kis- és nagybetűkre! Ezzel a rendszerszintű beállítások végére is értünk, az ez után következő beállítások az adatfolyamra vonatkoznak majd.

- `StreamTitle`: ez az adásunk címe, az itt beállított érték látszik a hallgatónál a ***cím*** rovatban, miután csatlakozott hozzánk.
- `StreamURL`: ez egy adathivatkozás, amely a hallgatónál szintén megjelenik, és ezen a weblapon biztosíthatunk számára egyéb adatokat az adásról.
- `Shuffle=0/1`: ha ennek az értéke nem nulla, akkor a lejátszási lista tartalmát nem a felsorolt fájlnevek sorrendjében játssza le, hanem véletlenszerűen választ a repertoárból.
- `BitRate/SampleRate/Channels`: ezekkel az értékekkel befolyásolhatjuk az elkészülő MP3-folyam jellemzőit, a minőségét, s ezzel együtt méretét. A `BitRate` mutatja meg, hogy egységnyi hosszon mennyi adatot tárolunk a zenéből, a `SampleRate` a mintavételezési frekvenciát jelenti, míg a `Channels` értéke a csatornák számát. Ez magyarán annyit tesz, hogyha a `Channels` értéke `1`, akkor a program mono-, s amennyiben az érték `2`, akkor sztereoadást készít. Attól függően, hogy milyen kapcsolatunk van és hol szeretnénk sugározni, állítsunk be valamilyen értéket. Ha helyi hálózaton (LAN) sugárzunk, célszerű a `128000/44100/2` (`BitRate/SampleRate/Channels`) hármast használni, lassabb kapcsolat esetén felezzük, negyedeljük az értékeket, s monoadatfolyamot sugározzunk.

Kapcsoljuk be a rádióállomást!

Ha ezekkel az alapbeállításokkal végeztünk, nincs más dolgunk, mint futtatni az adatfolyam-készítő programunkat, amely az adást továbbítani fogja a kiszolgálónak, amihez már csatlakozhatnak is a hallgatók. Jelen esetben nekünk az `sc_trans_linux` nevű binárist kell futtatnunk. Elindítás után láthatjuk, hogy csatlakozik a másik szálon futó kiszolgálóhoz. A másik szálra átkapcsolva máris láthatjuk, hogy a kiszolgáló működik, s azt is megmutatja, hogy hány hallgató csatlakozott hozzá. Ha semmilyen hibaüzenetet nem kaptunk, az azt jelenti, hogy a gépünk máris rádióállomás. Ellenőrzésképpen indítsunk el egy XMMS-t, és fájl helyett most a ➲ ***http://localhost:8000*** URL-t adjunk hozzá a lejátszási listához, amely a saját gépünkön futó SHOUTcast kiszolgáló címe. Ezek után hallanunk kellene a hangszórókból kiáramló muzsikát, a zene címében pedig az általunk beállított rádióadás címét kell látnunk. Ha valóban működik, a kezdeti sikereken felbuzdulva próbáljunk meg más gépekről is csatlakozni a kiszolgálóhoz, s élvezzük az éterben megszületett új lehetőségeket. Miután egy kicsit megnyugodtunk, vizsgáljuk meg a hiányosságokat is. Tegyük fel, hogy folyamatos adást szeretnénk, ugyanakkor változtatásokra van szükség a lejátszási listában: másik számot szeretnénk elindítani, esetleg jó lenne, ha a programunk véletlenszerűen, és nem sorban játszaná le a zenéket stb. Mivel azonban a TRANScast a beállításokat csak indításkor olvassa be, első megközelítésben a lejátszási lista vagy a beállításfájl átírása után újra kellene indítanunk a programot, amely

A kiszolgáló indítása

Az ügyfél indítása

mindenképpen egy kis szünetet okozna az adásban – arról nem is beszélve, hogy a kiszolgáló bizonyos idő eltelte után alapértelmezetten eldobja a hallgatók által létesített kapcsolatokat. Hogyan lehetne ezt másképp megoldani?
A válasz egyszerű: a fejlesztők is gondoltak erre a problémára, ezért lehetővé tették számunkra, hogy éppen futó TRANScast-folyamatunknak jelek (signals) formájában üzenetet küldjünk.

A TRANScast kezelése jelekkel

A programot úgy írták meg, hogy futás közben lekezeljen bizonyos üzeneteket, amelyek segítségével a fent említett módosítások elvégezhetők. Ahhoz, hogy ilyen jelet küldjünk a programnak, szükségünk van az indítása során a rendszermagtól kapott folyamatazonosítóra (PID), amit indítás után közvetlenül ki is ír a konzolra. Vegyük például az alábbit:

```
<10/27/03@10:36:06>
➥[MAIN] PID: 2203)
```

Ezek után a `kill` paranccsal a folyamatnak máris olyan üzeneteket küldhetünk, amelyeket kezelni tud, ehhez azonban előbb ismerjük meg ezeket a bizonyos jeleket:

- `HUP`: hatására a program frissíti a naplófájlokat, közben a konzolra történő naplózás leáll.
- `WINCH`: ha ilyen jelet küldünk a programnak, azonnal elkezdi játszani a lejátszási lista következő számát.

A SHOUTcast webes felülete

- `USR1`: újraolvassa a lejátszási listában szereplő elemeket, de eközben a jelenlegi fájl lejátszását nem szakítja meg, ám amikor az véget ér, már az új lejátszási lista elemeit fogja játszani.
- `USR2`: ki-, illetve bekapcsolja a véletlenszerű lejátszási sorrendet. (Ha ki van kapcsolva, a lejátszási listában megadott sorrendnek megfelelően játssza le az elemeket.)
- `TERM`: leállítja a TRANScast adatfolyam-kiszolgálót.

Ezek után, ha a lejátszási listához újonnan hozzáadott zenéket szeretnénk beépíteni az adásba, adjuk ki a '`kill -s USR1 9999`' parancsot, ahol a 9999-es PID helyére mindenki a TRANScast indításakor kijelzett négyjegyű folyamatazonosítót írja. Fontos, hogy lehetőleg ugyanannak a felhasználónak a nevében adjuk ki a parancsot, aki másik szálon a programot is futtatja, különben azt a hibaüzenetet kapjuk, hogy nincs jogunk jelet küldeni a programnak. Ha semmilyen üzenetet nem kapunk a `kill` parancs végrehajtása után, az azt jelenti, hogy a jel eljutott a címzetthez, s az ezt le is kezelte. Ennek ellenőrzéséhez váltsunk át a futó TRANScasthoz, a s a konzolra azt kell kiírva látnunk, hogy az általunk kiadott parancsot az alábbi eredménnyel fogadta:

```
<10/29/03@11:39:46> [MAIN]
➥SIGUSR1; Reload Playlist
<10/29/03@11:39:46> [MAIN]
➥Reloading playlist
<10/29/03@11:39:46> [MAIN]
➥Loading playlist
(example.lst)
<10/29/03@11:39:46> [MAIN]
➥Found (3) entries in playlist
```

Javaslom, hogy az összes jelet próbálgassuk ki, figyelgessük a hatását olyan módon, hogy közben a saját rádióadásunkat hallgatjuk.

A SHOUTcast kiszolgáló mint webkiszolgáló

Az előző cikkben is említettem már, hogy a legtöbb adatfolyam-szolgáltatás a HTTP protokoll felett zajlik, ami azt jelenti, hogy a rádióállomásoknak valójában egy webkiszolgálót kell megvalósítaniuk. A SHOUTcast kiszolgáló esetében is ugyanez a helyzet, s a fejlesztők, ha már úgyis egy csökkentett tudású webkiszolgálót írtak, gondoskodtak arról, hogy egy böngésző segítségével is csatlakozhassunk a kiszolgálóhoz. Ám ekkor nem az adatfolyamot kapjuk, hanem mindenféle többletadatot a rádióállomással, az épp futó adással, a hallgatókkal stb. kapcsolatban. Ezenkívül rendszergazdai feladatok ellátására is lehetőségünk nyílik (bizonyos IP-k letiltása, bizonyos hallgatók kirúgása stb.). A szolgáltatás igénybevételéhez nem kell mást tennünk, mint a böngészőnkben beírni a rádióállomás címét, kapuszámát. Futtassuk még mindig a kiszolgálónkat, a TRANScast segítségével küldjünk neki adatfolyamot, s XMMS segítségével a gépünkről csatlakozzunk saját magunkhoz – a lényeg, hogy legyen működő kiszolgáló, s hozzá valamekkora (akár egy főnyi) hallgatóság. Ekkor indítsunk egy böngészőt, és látogassunk el a ➲ ***http://localhost:8000*** címre. Máris láthatjuk rádióállomásunk jellemzőit. A jobb oldalon található ***Admin login*** menüpontra kattintva az `admin/<beállított kiszolgáló jelszó>` párossal azonosíthatjuk magunkat, s elénk tárul a beavatkozó felület, amelynek segítségével felügyelhetjük a rádióállomást. A SHOUTcast kiszolgáló beállításai között megadhatjuk, hogy ez a felület csak kivételes esetben engedjen meg beavatkozást, általánosan inkább csak adatokat jelenítsen meg.

Áttekintés

Gondolom, senki sem vitatja a megoldás egyszerűségében rejlő nagyszerűségét. Természetesen az egyszerűség alatt nem feltétlenül a sok-sok beállítási lehetőséget értem, hanem azt, hogy pár óra alatt rádióadást vagyunk képesek csinálni! Hol lehet mindez hasznos? Nem szeretnék az élettől távol eső példákkal szolgálni, ezért álljon itt egy ellesett ötlet: képzeljünk el egy kollégiumot, ahol több száz diák lakik, s a közösségi életet egy mindenki által élvezhető rádióadás beindítása színesítené. A legtöbb kollégiumban mára már kiépítették a helyi hálózatokat, amelyek sávszélessége nagyságrendileg 100 Mbit körül mozog. Ez kényelmesen elég ahhoz, hogy a belső hálózaton hozzuk létre a közösség rádióállomását, amelyhez minden – az adott tartományban lévő – gép csatlakozhat. (Gondoljunk most bele abba, hogy egy hagyományos, kis hatósugarú állomás üzemeltetéséhez mindenféle, általában pénzes engedélyek szükségesek, a „kábelen" terjedő adás pedig a kábelezés iszonyatos költségét ruházná ránk.) Mire van mindehhez szükségünk? Egy Pentium II-es processzorral szerelt gépre, erre az ingyenesen használható adatfolyam-kiszolgálóra – és némi rátermettségre, hogy mindezt megvalósítsuk! Ahhoz, hogy ez a szolgáltatás kellő színvonalú legyen, természetesen nem elég ez az egyébként működő összeállítás: nem árt, ha kiszolgálónkat teljesen testreszabjuk. Ehhez szeretne segítséget nyújtani sorozatunk következő része, amelyben az emelt szintű beállításokról és a webes felület használatáról szeretnék szólni.

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

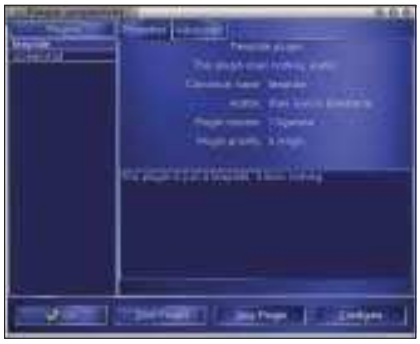
A Creative Webcam telepítésének kálváriája

Régóta nem hagyott nyugodni a kérdés, hogy miként is lehetne webkamerával felszerelkezve kommunikálni az interneten. Arról nem is beszélve, hogy egy ilyen pici eszköz ugyan nem váltja ki a ma használatos digitális fényképezőgépeket, de megkönnyítheti az életünket.

Igyekeztem körbenézni, körbekérdezni linuxos ismerőseimet, hogy mely típusok használhatóak jól Linux alatt. Nos, sajnos szinte semmilyen használható útmutatáshoz nem jutottam. Ennek oka valószínűleg a csekély linuxos támogatás és a webkamerák kis mértékben való meghonosodottsága.
Az internet hosszú, napokig tartó böngészése során is csak annyi adatra tettem szert, hogy az OV511 lapkával felszerelt kamerák jól működnek Linuxon. Természetesen ezek is USB-kaput használnak. Végül arra a következtetésre jutottam, hogy elvben a Genius VideoCam Express és a Creative Webcam Plus is működik Linuxon.
Ennyi óvatosságból fakadó utánajárást követően vettem egy Creative Webcamet – nem „Plus”-ost, csak egyszerűen egy Creative Webcamet. Kellemetlen élmény ért, amikor a boltban a linuxos tapasztalatok felől érdeklődtem – azt sem tudták, hogy mi fán terem a Linux. Mielőtt bárki rosszindulattal vádolna az ezek után kialakult elmarasztaló véleményem miatt, azért lássuk be, hogy 2003-ban egy szaküzletben a Linux-ismeretnek már természetesnek kellene lennie.
Témánkhoz visszakanyarodva: kétségek között hagyva, de boldogan vittem haza a webkamerát, amit igazából alig több mint 6000 forintért vettem meg. Nagy siettemben szinte széttéptem a csomagolását, majd rádugtam az első USB-kapura, amit felleltem a gépen, ezután bekapcsoltam a számítógépet.
Nagy örömmel láttam UHU-Linuxomon, hogy felismerte, és minden USB-modult betöltött. A KDE vezérlőpultja is felismerte, kiírta a típust, a lapkát… szerinte azonban a lapka nem OV511-es, hanem OV518-as volt. Ellenőriztem, hogy létrejött-e a */dev/video0* eszközfájl, majd elindítottam a már előre beszerzett és lefordított `xawtv` 3.88-ast. Ekkor sem történt semmi, konzolból futtatva azt írta ki, hogy ilyen eszköz nincs.
Három nap szükségeltetett ahhoz, hogy rájöjjek, mi hogyan is működik. Következzék hát az a leírás, amelyben azért számolok be a tapasztalataimról, hogy másnak már ne kelljen ennyit bajlódnia a webkamera telepítésével.
Sokáig keresgéltem rá az interneten az OV511, illetve OV518 kulcsszavakra, míg végül találtam egy nagyszerű hivatkozást. Ezen az oldalon kifejezetten ezekhez a kamerákhoz írt meghajtók készítői tették közzé a munkáikat (➲ http://alpha.dyndns.org/ov511). Több rendszermaghoz több változatszámú meghajtó létezik, például az 1.xx-esek és a 2.xx-esek.
Végigböngésztem a támogatott kamerák listáját (érdemes az egészet kinyomtatni), meg is találtam közöttük az enyémet. A kamera összes adata fontos:

- név: Creative Webcam;
- lapka: ov518;
- SN: PD1001;
- sensor: OV6620.

Az SN-adat könnyen ellenőrizhető a kamera hátulján, az érzékelő (sensor) és a lapka adatát pedig maga a rendszer közli velünk az USB-kapun vett adatok alapján.
E kamera támogatottsági listáján olvashatjuk, hogy csupán a 2.22-es vagy a frissebb meghajtókat támogatja.
Egy csomó nézelődés és próbálgatás után végül rájöttem, hogy a legfrissebb 2.25-ös meghajtó hibás, vagy legalábbis én nem igazán tudtam használni.
Szerintem a 2.23-as vagy a 2.24-es változatot érdemes letölteni. A kicsomagolás után a 2.23-ast fordítottam le.
Ne lepődjünk meg azonban, ha az `ov518_decomp.o` modulnál hibát jelez. Én is sokáig azt hittem, hogy ez valóban hiba, de figyelmen kívül hagyható, mert mint a GYK-ból megtudtam, az 2.xx-es meghajtósorozatban ezt a modult nem szükséges betölteni. Ha ezt kézzel próbáljuk meg elkövetni, észre fogjuk venni, hogy a rendszer állandóan „kiköpi”. Tehát nyugodtan hagyjuk figyelmen kívül, mert hiába fordítjuk



le külön, nem fogjuk tudni betölteni.
Miután a meghajtó lefordul és a helyére kerül, érdemes végigkövetni a forráskód könyvtárában található *ov511.txt*-ben leírtakat. Itt a következőket olvashatjuk: a fordítás után egyenként töltsük be a modulokat a `modprobe` paranccsal. Ezek egy részét egy korszerű Linux-rendszer, mint amilyen az UHU, és mint később látni fogjuk, a SuSE 8.2-es is – betölti.
A leírásban szereplő `usb-core` és `usb-uhci` kézi betöltésére általában semmi szükség, de azért nem árt ellenőrizni. Ami viszont utána következik, arra annál inkább szükségünk lesz.
A folyamat a következő lépésekből áll:

1. Jelentkezzünk be rendszergazdaként a terminálba: `su -`
2. Egymás után adjuk ki a következő utasításokat:
   ```
   modprobe videodev
   modprobe i2c-core
   modprobe ov511
   ```
   Olvassuk el figyelmesen az *ov511.txt*-t, ugyanis számos kapcsolóval testre-

szabható. Figyelem, ha úgy érezzük, hogy elrontottuk a paraméterezést, vagy hibás értéket adtunk meg, az `rmmod ov511` paranccsal „kilőhetjük” és újra betölthetjük a modult.

3. `modprobe ovsensor`
   Ez is nagyon fontos modul, ügyeljünk rá, hogy ez a lépés ki ne maradjon.
4. Ezek után felhasználóként adjuk ki az `xawtv` utasítást. Ha mindent jól csináltunk, a `xawtv` program, ami az ov5xx-es kamerákat támogatja, gond nélkül elindul.

A puding próbája

Nekiveselkedtem a kipróbálás izgalmas feladatának. A sebességgel nem volt gondom, hozta a 30 f/s körüli gyári értéket, a felbontás is megfelelt 352×288 16 biten. A jobb képminőség eléréséhez a monitort érdemes erre a színmélységre állítani. Viszont a `xawtv`-vel készített fotó (snapshot) mindig olyan sötétre sikeredett, hogy vagy egy nagy fekete lyuk lett a képből, vagy amikor iszonyatosan túlvilágítottam magam, esetleg egy-két arcvonásom is felismerhető lett. Másfél nap kínlódás után feladtam, és ugyanezt megpróbáltam a SuSE 8.2 alatt is. A fenti folyamatot ismételtem meg, és az eredmény is ugyanaz lett: sötét, nézhetetlen kép. Nem sikerült rájönnöm, mi lehetett a jelenség oka, az alábbiak miatt a `xawtv` hibájára gyanakszom. Webkamera témakörben kerestem írásokat, többek között az eddigi Linux-világokban is, a 2003. februári szám 70–71. oldalán rá is akadtam a GnomeMeeting videokonferencia-programra. Rákerestem a SuSE telepítőjében, majd három perc múlva már a gépemen virított a program. (UHU alá le kell fordítani, de `.uhu` csomagban is megtalálható, és az `apt-get`-tel telepíthető.) A képminőség jobb, a színeket is jobban kezeli, ráadásul a fotó (snapshot) szinte megegyezik azzal a képpel, amit a videoablakban látunk. Hátránya, hogy nem tudunk vele videót rögzíteni, bár lehet, hogy csak én nem jöttem rá, hogyan kell. Az előzmények ismeretében azonban ez nem is olyan végzetes veszteség, hiszen a `xawtv` még képet sem tudott készíteni, nemhogy videót. Jelenleg ehhez a kamera- és meghajtótípushoz a GnomeMeetinget vagy a Zappingot ajánlom. Megjegyzés: a GnomeMeeting nagyon jó program, de fotó vagy videó rögzítésére azért akad jobb is. Nekem tökéletesen elindult a Zapping gnome-os tévéprogram. Mivel tévékártyám és két */dev/videoX* eszközleíróm is van, mindkettővel kipróbáltam. Figyelem, az UHU először mindig a tévékártyát fogja társítani, tehát hiába került először a gépre az USB-s webkamera, az utóbb berakott tévékártya lett a */dev/video0* és a webkamera a */dev/video1*!

A Zapping indítása ilyenkor azonban egy kicsit másként zajlik, mint ahogyan azt megszokhattuk. Ha két eszközünk van, akkor célszerű értéket adni a programnak:

```
zapping -device=/dev/videoX
```

Ha az X értéke 1, akkor a webkamerát használja, ha pedig 0, akkor a tévékártyát. Figyelem, ez csak az UHU-Linuxra vonatkozik, mert SuSE-ban megtartja a telepítési sorrendet, vagyis a */dev/video1* lesz a tévékártya és a */dev/video0* a kamera. A Zapping további hasznos képességei: jó fotót készít és videót is rögzít, sőt ha mikrofonunk is van, akkor mindezt hanggal teszi. Ráadásul rögtön MPEG-be képes kódolni a videót – ehhez a művelethez azonban egy legalább 700–800 MHz-es gép ajánlott, mert az én 525-re húzott Celeronom alatt olykor képtelen minden képkockát rögzíteni (frame drop).

A modulok betöltése nélkül egyáltalán nem fog működni a kamera, ráadásul minden indítás után rendszergazdaként, kézzel kell őket betölteni. De vajon valóban minden indítás után el kell ezt játszanunk? A válasz: egyértelműen nem. Okos megoldás, ha a feladatot önműködővé tesszük, tehát olyan módon próbáljuk megoldani a feladatot, hogy az indítás során mindent töltsön be.

A megoldás pofonegyszerű: a `modprobe` *`modulnév`* sort rakjuk be a */etc/inid.d/system/boot* parancsfájlba (ezt csak UHU-Linux alatt próbáltam ki), méghozzá a végére, közvetlenül az `exit0` sor fölé.

Ezután a parancsállomány így fest:

```
modprobe videodev
  modprobe i2c-core
  modprobe ov511
# Választható érték, ha
# akarjuk, megadhatjuk,
# de nem szükséges.
  modprobe ovsensor
# Le ne hagyjuk, mert ez
# a képérzékelő meghajtója!
  exit0
```

A Zapping kapcsán érdemes még elgondolkodni azon, mit is ajánlatos beállítanunk. Mindenképpen képminőség-javulást tapasztalunk, ha a kamera alapértelmezett 352×288-as legnagyobb felbontását elfelejtjük, és a kép-, illetve videorögzítés során kisebb értéket adunk meg neki. Ehhez javaslom a ma már szinte mindenki által ismert 320×240-es felbontást, ami különösen akkor segíthet, ha lassúbb gépen szeretnénk dolgozni a kamerával. Ekkor az is javít a sebességen, ha a KDE helyett mondjuk az IceWm-et használjuk. Saját tapasztalatom szerint szemmel látható a képminőség-javulás.

Érdemes még kitérni a Zapping kodekjeire, amivel videót is rögzíthetünk. Ekkor különösen fontos lesz, hogy milyen gépen szeretnénk felvenni az anyagot. Tekintélyes listát találunk a programban erre vonatkozólag, ***.avi*** és ***.mpeg*** kiterjesztésű filmekhez. Hogy teljes legyen a kényelem – és a gépre méretezhető az erőforrás-használat –, állíthatjuk a bitsebességet, és a használt kodek által rögzített kép felbontását. Következzenek a száraz adatok! Nem mindegy ugyanis, hogy a feladat menynyi erőforrást von el a géptől. Jelenlegi gépem kiépítése a következő: Celeron 466 (most 525 MHz-en dorombol), 320 MB SDRAM, GeForce2 MX videokártya, SoundBlaster 16 Vibra hangkártya, 2×20 GB Maxtor merevlemez (7200 rpm), DVD/CDRW kombó és két Realtek hálózati kártya. Természetesen nem maradhat el a többi szabványos alkatrész sem: a 15"-os monitor, PS/2-es egér és billentyűzet. Erre a gépre csatlakoztattam a webkamerát. Pillanatnyilag UHU-Linux fut rajta, KDE 3.1-es felületen OpenOffice.org 1.0.2-es, a GnomeMeeting, a Ksirc, a Konqueror és a Kmail társaságában. Ezen a kiépítésen ilyen terhelés mellett a GnomeMeeting folyamatosan közvetítette a képet, és a processzorkihasználtság sosem megy húsz százalék fölé. Ez az érték fotózás közben ugrik fel egy pillanatra 37–40 százalék közé. Összehasonlításképpen: a `xawtv`-re 60–65 százalékos alaperőforrás-használat jellemző, mely érték a (bűnrossz) kép készítésekor 72–78 százalékra is felugrik. A méréshez a `top` parancsot használtam.

Sok sikert a kamera telepítéséhez!

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.

Egységes és objektumközpontú adatbázis-kezelés

A Qt könyvtár adatbázis-kezelő alrendszerének bemutatása.

Néhány éve még irigykedve néztük a Windows (ADO, BDE) és Java (JDBC) környezetek által megvalósított adatbáziskezelő-független és objektumközpontú programalrendszereket. Természetes programozói igény, hogy a forráskód egyik adatkezelő környezetből könnyen átvihető legyen a másikba. Vajon miért? Lehetséges, hogy idővel nagyobb tudású adatbázis-kezelő környezetre lesz szükségünk, de az is gyakori, hogy a programot használók különféle adatbázismotorokat szeretnének használni. A nyílt ODBC szabvány régóta létezik Linuxra is, de ennek több komoly hátránya is akad, amelyek közül most csak kettőt említünk meg:

- Az ODBC nem objektumközpontú API.
- Az ODBC és a grafikus GUI-elemek adattartalmainak az illesztése nehézkes. Ez azt jelenti, hogy az adatforrások (táblák, nézetek) és a megjelenítésvezérlők együttműködése nem valósul meg önműködően egy adatforrásban szereplő összekötő objektumon keresztül.

Ez a helyzet mára már gyökeresen megváltozott. A Qt/KDE és Gtk+/Gnome rendszerek egyaránt fejlett, objektumközpontú API-modulokkal rendelkeznek. A Gnome környezet a GNOME-DB alrendszert bocsátja rendelkezésünkre, míg a Qt/KDE a Qt SQL-modult. Mindkét programozói környezet kiváló minőségű és teljes. Ebben a cikkben most a Qt/KDE SQL-környezetet szeretném bemutatni.

A Qt adatbázis-kezelő API felépítése

A Qt SQL-modul három programrétegből áll. A legalsó az illesztőréteg, ami az adatbáziskezelő-függő részeket tartalmazza, és egy felületet biztosít az adatbáziskezelő-független SQL API réteg számára. Az illesztőréteg jól meghatározott szabályok szerint épül fel, így egy új adatbázis-kezelő rendszer összevonása a Qt SQL-modulba könnyen megvalósítható. A legismertebb adatbázis-kiszolgálók illesztőit a Qt-csomag tartalmazza:

```
ORACLE (QOCI8)
PostgreSQL (QPSQL7)
MySQL (QMYSQL3)
MS SQL szerver és Sybase (QTDS7)
UNIX ODBC használata (QODBC3)
```

A fenti felsorolásban zárójelben adtuk meg azokat a karakterlánc-szimbólumokat, amelyek az adott illesztő indítását kérik. A SQL API réteg felépítése hasonló más hasonló API-k szerkezetéhez. Tekintsük át röviden az ebben a rétegben lévő adatelérő és -kezelő osztályokat:

- `QSqlDatabase` osztály: a kapcsolatosztályt (`session`, `connection`) valósítja meg. Ennek az osztálynak az objektumai teszik lehetővé például az adatbázishoz való kapcsolódást, illetve a tranzakció-kezelést.
- `QSqlQuery` osztály: egy tetszőleges SQL-parancs (`select`, `insert`, `update`, `delete`, `create...`, `alter...`) kiadását teszi lehetővé. Lekérdezés esetén az eredményhalmaz kezelését is támogatja.

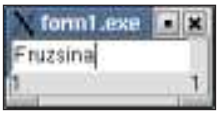
1. kép A táblázat1.exe program futási képe

- `QSqlCursor` osztály: a táblák, nézetek soraiból összeállított klasszikus SQL-kurzor kezelését valósítja meg. Az eredménytábla adataihoz való hozzáférés mellett az SQL-kurzoron keresztül történő adatmódosító műveleteket (`insert`, `update`, `delete`) is támogatja. Ennek az osztálynak van egy nagyon fontos további szerepe is, ugyanis adatforrásul működik a GUI-vezérlők számára.
- `QSqlRecord` osztály: egy tábla vagy nézet egy sorát képviseli, illetve a sornak megfelelő adatmezők gyűjteményét tartalmazza. Lehetővé teszi a sor adatainak (adatmezők) lekérdezését és módosítását.
- `QSqlField` osztály: egy tábla vagy nézet egy-egy adatoszlopának a kezelését tudja.
- `QSqlError` osztály: az SQL-műveletek során fellépő hibákról ad részletesebb tájékoztatást.
- `QSqlIndex` osztály: az adatbázisindexek és a kurzorok rendezettségét támogató osztály.

A Qt SQL alrendszer utolsó programrétege a felhasználói illesztőréteg, vagyis a grafikus elemek (data-aware widgets) – nagyrészt ezek végett használjuk a Qt könyvtárat. Itt jegyezzük meg, hogy a Qt SQL használatához nem szükséges ennek a szintnek az igénybevétele, azaz karakteralapú (`ncurses`, `tvision`) programokat is készíthetünk a Qt SQL API használatával. Ezek a vezérlők önműködően kezelik adatbázisunk adatait, ebben a szerkezetben a `QSqlCursor` osztálybeli objektum tölti be az adatforrás szerepét. Tekintsük át e réteg fontosabb osztályait:

- `QDataBrowser` osztály: ennek az osztálynak a használatával adatbevivő űrlapok készíthetők, illetve az adatbázisban böngésző űrlapok készítését is támogatja.
- `QDataTable` osztály: szerepe hasonló a `QDataBrowser` osztályéhoz, de a táblázatos megjelenítést támogatja.

- `QDataView` osztály: amennyiben csak olvasható űrlapot szeretnénk készíteni, úgy ebből az osztályból érdemes építkezni. Kinézete és működése a `QDataBrowser` osztályéhoz hasonló.
- `QSqlForm` osztály: ez az osztály teszi lehetővé és kezeli azt, hogy az általános Qt-vezérlőkből adatbázis-kezelő űrlapokat építhessünk.
- `QSqlPropertyMap` osztály: ennek az osztálynak egy-egy objektuma hordozza azt az adatot, amely megmondja, hogy a GUI-vezérlők mely adatbázistábla melyik mezőjéhez vannak rendelve. Ki szeretnénk emelni, hogy e cikk készítése során végig a PostgreSQL adatbázis-kezelőt használtuk.



2. kép A form1. exe

Hogyan készíthetünk Qt alapú adatbázis-kezelő programokat?

Legyen egy `cs_adatok` nevű PostgreSQL adatbázisunk, amiben egy névnapokat tartalmazó `nevek` nevű tábla szerepel. A tábla szerkezete egyszerű, három oszlopa van:

```
honap - number
nap - number
nevnap - varchar(50)
```

A feladatunk az, hogy szöveges üzemmódban maradva listázzuk ki a képernyőre a tábla tartalmát. Íme a feladatot megoldó C++-forráskód (*1. lista*, lásd még az 54. CD-n a Magazin/Qt/sql_1.cpp-t):

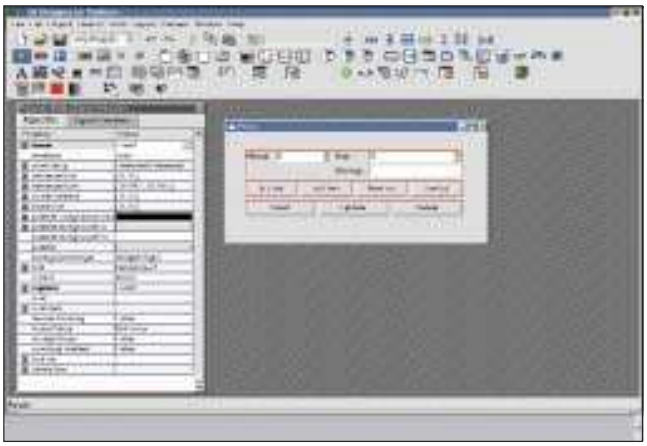
A 14. sorban az `app` objektumot csak létre kell hozni, semmilyen további teendő nincs vele. A `QApplication` osztály létrehozójának 3. kapcsolója azt vezérli, hogy grafikus vagy karakteralapú programot szeretnénk-e készíteni. Jelen esetben ennek a kapcsolónak `false` értéket adtunk, ami a szöveges alapú alkalmazások készítését teszi lehetővé. Ez a kapcsoló alapértelmezésben `true`, azaz ekkor a Qt grafikus összetevőit is használhatjuk. A 15. sorban lévő `pg` mutató egy adatbázis-kapcsolatot jelöl. Látható, hogy a `QSqlDatabase` osztály statikus `addDatabase()` tagfüggvényét hogyan láttuk el értékkel a `QPSQL7` karakterlánccal, azaz megmondjuk, hogy a `pg` mutatón keresztül egy PostgreSQL adatbázist szeretnénk majd kezelni, ami egy adatbáziskezelő-illesztőigénylést is jelent.
A 19–23. sor a kapcsolódáshoz szükséges bejelentkezési adatok kitöltését szemlélteti, amit természetesen a `pg` által mutatott objektum fog tartalmazni. Érdekességként megjegyezzük, hogy Oracle-adatbázis esetén a `setDatabaseName()` tagfüggvény értéke a `TNS` név. A kód 25. sorában megpróbálunk az adatbázishoz kapcsolódni. Az ügyfél-kiszolgálókapcsolat felépülése után a 27. sorban létrehozunk egy `q` SQL lekérdezést lehetővé tevő objektumot olyan módon, hogy a létrehozójában mindjárt meg is adjuk a végrehajtandó SQL-parancsot, amely még a létrehozóban végre is hajtódik. A 28. sor `isActive()` tagfüggvénye akkor lesz igaz, ha az SQL-lekérdezés eredménytáblája sikeresen létrejött.
A 30–35. sorban egy ciklusban kiírjuk a lekért tábla összes sorát a képernyőre. Az eredmény táblában a `next()` tagfüggvény áll a következő sorra, amit működő sornak is nevezünk. A `q` objektum `value()` tagfüggvénye teszi elérhetővé a lekért adatokat. A program befejezése előtt a 38. sorban lezárjuk az adatbázis-kapcsolatot.
Ennyi magyarázat után tekintsük át az ***sql_1.cpp*** program fordítását! Érdemes egy `Makefile`-t készíteni hozzá a következő tartalommal, amelyben a `-I` és `-L` utáni útvonal a Qt telepítési helyétől függően eltérő is lehet:

1. lista Az sql_1.cpp forráskódja

```
1. //
2. // sql_1.cpp
3. //
4. #include <iostream>
5. #include <qapplication.h>
6. #include <qsqldatabase.h>
7.
8. using std::cout;
9.
10.//--- A program indulási pontja ---
11.int main(int argc, char **argv)
12.{
13.
14. QApplication app(argc, argv, false);
15. QSqlDatabase *pg =
    ➥QSqlDatabase::addDatabase( "QPSQL7" );
16.
17. if ( !pg ) { cout << "DRIVER hiba!\n";
                          ➥return 1; }
18.
19. pg->setDatabaseName("cs_adatok");
20. pg->setUserName("postgres");
21. pg->setPassword("111111");
22. pg->setHostName("localhost");
23. pg->setPort( 5432 );
24.
25. if ( !pg->open() ) { cout << "Hiba az
    ➥AB nyitáskor!\n"; return 1; }
26.
27. QSqlQuery q("select * from nevek;");
28. if ( q.isActive() )
29. {
30. while ( q.next() )
31. {
32. cout << "\n" << q.value( 0 ).toString()
         ➥<< "\t";
33. cout << q.value( 1 ).toString() << "\t";
34. cout << q.value( 2 ).toString();
35. }
36. }
37.
38. pg->close();
39.
40. return 0;
41.}*
```

```
CXFLAGS = -I/usr/lib/qt3/include
➥-L/usr/lib/qt3/lib
LIBS = -lqt-mt
sql_1.exe : sql_1.cpp
g++ $(CXFLAGS) -o sql_1.exe $(LIBS) sql_1.cpp
```

Ezek után adjuk ki a `make` parancsot, ami elkészíti az ***sql_1.exe*** bináris programot, amit ha végrehajtunk, láthatjuk táblánk végiggördülő tartalmát a képernyőn.
Megjegyzés: programunkban természetesen több `QSqlDatabase` objektum is lehet, hiszen néha olyan a feladatunk, hogy több – akár különböző típusú – adatbázisból kell



3. kép A QT Designer

2. lista Általános SQL-parancsok

```
q.exec("create table ujnevek as select
➥* from nevek;");
q.exec("alter table ujnevek add constraint
➥pk_ujnevek primary key(honap,
nap);");
q.exec("select * from ujnevek;");
q.exec("insert into ujnevek (honap, nap,
➥nevnap) values ('1', '1', 'Linus')" );
```

3. lista Tranzakció-kezelés

```
1. if ( pg->transaction() )
   // Egy tranzakciót kezdünk, ha tudunk
2. {
3. q.exec("insert into ujnevek (honap, nap,
   ➥nevnap) values ('1', '1', 'Linus')" );
4. pg->commit();
5. }
6. else
7. {
8. cout << "\nA tranzakció megkezdése
   ➥nem sikerült!";
9. }
```

egyszerre dolgoznunk. Ilyenkor az adatbázis-kezelő műveleteket megvalósító tagfüggvények egyik értékeként mindig jeleznünk kell, hogy melyik munkamenetre irányul a kérés.

Az általános SQL parancskezelő osztály

A `QSqlQuery` osztály egy általános SQL parancsfogadó szolgáltatást valósít meg. Amennyiben a `pg` mutató egy érvényes adatbázis-kapcsolatot ír le, akkor a következő változómeghatározást is használhatjuk:

```
QSqlQuery q("", pg); // A q a közvetlen SQL
parancsok végrehajtója
```

Látható, hogy a létrejött `q` objektum a `pg` által megvalósított adatbázisra vonatkozóan fog tudni SQL-parancsokat elküldeni. A létrehozó első értékében azt is jeleztük, hogy nem kérünk azonnali parancsvégrehajtást. A `q.exec()` tagfüggvény szolgál a parancsok végrehajtására, ezek bármilyen érvényes SQL-utasítások lehetnek, (lásd például a *2. listát*).
A `q` objektum a `select` lefutása után magát az SQL-kurzort, azaz az eredménytáblát és annak adatelérését is tartalmazza. Ennek megfelelően léteznek például az eredménytábla sorain mozgó `first()`, `next()`, `last()`... tagfüggvények. Az `at()` tagfüggvény visszaadja, hogy az aktív sor hányadik (az indexelés 0-tól kezdődik). A `seek()` tagfüggvény még a közvetlen pozicionálást is lehetővé teszi. A `size()` az eredménysorok számát adja vissza. Megjegyezzük, hogy a C nyelv `sprintf()` függvénye segítségével könnyen létre tudunk értékekkel megtűzdelt SQL-parancsokat hozni, például:

```
char sqltxt[200];
sprintf(sqltxt, "select * from nevek where
➥honap=%d and nap=%d", 11, 5);
```

Tranzakció-kezelés

A tranzakció-kezelés természetesen a `QSqlDatabase` osztály (a munkamenet, azaz session) objektumán keresztül valósul meg. Legyen a `pg` egy ilyen osztály objektumára mutató változó. Ekkor a tranzakciót kezelő kód jellemzően ilyen szerkezetű lesz (*3. lista*).
Az 1. kódsor `pg->transaction()` tagfüggvénye kijelöli a tranzakció kezdetét. Lehetséges, hogy az általunk használt adatbázis-kezelő nem tud tranzakciókat kezelni, ezt az esetet a 6–9. sorok kezelik le.
A 3. sor `insert` művelete után a `pg->commit()` és a `pg->rollback()` hívást is választhatnánk, azaz véglegesíthetjük a változtatásokat, illetve el is vethetjük őket. Esetünkben a 4. sorban egy `commit()` véglegesíti az új rekord beszúrását.
A tranzakció-kezelést az 54. CD Magazin/Qt/sql_2.cpp program használatával próbálhatjuk ki.
Sorozatunk következő részében a Qt alapú SQL kurzorok használatát, a hibakezelést, valamint az adatbázis-kezelő GUI készítési Qt lehetőségeket fogjuk áttekinteni. Szó lesz az űrlap (form) alapú alkalmazások készítésének módszeréről, de röviden kitérünk a Qt Designer használatára is.

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

A Blender kihasználása (2. rész)

Utolsó teendőként készítsük el játékunkat a Blenderben.

Mindenekelőtt fel szeretném hívni a figyelmet a CD-mellékleten található programra, amelynek segítségével a Blender 2.23-as változatában létrehozott objektumainkat szöveges formába menthetjük. Korábban már szóltam az UV koordináták mentésére készített programról, ennek továbbfejlesztését most olvasóink is használatba vehetik. A program alkalmas arra, hogy mentse a jelenetben található tárgyak pontjait, síklapjait, normálvektorait és természetesen a textúrakoordinátákat is. A forráskód elején megjegyzésben látható a kimeneti fájl formátuma.
A program a fájlokat a `defpath` változóban megadott könyvtárba menti. Használatához először be kell töltenünk a SHIFT-F11 billentyűk hatására előbukkanó szövegszerkesztőbe, majd az ALT-P billentyűkombinációval elindíthatjuk. Amennyiben sikeresen lefutott, a meghatározott könyvtárban megtaláljuk a mentett objektumokat, amelyeket későbbi felhasználás céljából tovább is alakíthatunk.

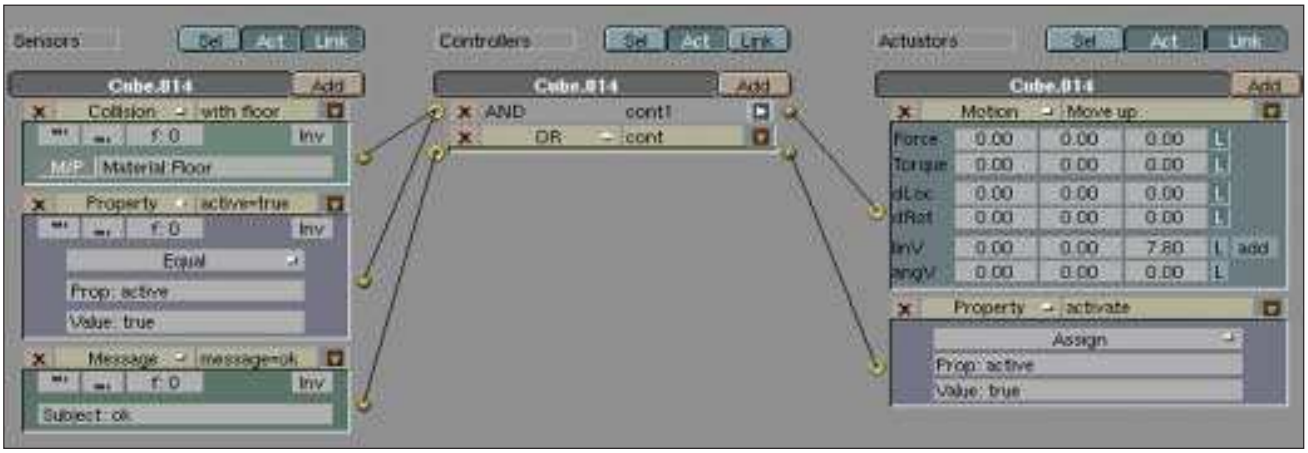
A játékterv

Ezek után kezdjük el a játék elkészítését. Egy-egy játék vagy bármilyen más program elkészítésének első lépése mindig a tervezés. Meghatározzuk, hogy mit kell tudnia a játéknak, hogyan kell működnie. Első egyszerű játékunk működése abból fog állni, hogy a játékos által irányítható figurát a pálya elejéről el kell juttatnunk a pálya túloldalán található kijárathoz, anélkül, hogy közben leesnénk a padlót képező síkról vagy valamelyik fel-le mozgó „kalapács” alá kerülnénk. A játékos a kurzormozgató billentyűkkel irányítható, és kapcsolatba kerülve az álló akadályok egyikével, azzal rugalmasan ütközik, majd arrébb pattan. A játékos dolgát nehezítik a pályán található fel-lemozgó kalapácsok, amikkel érintkezve a figura életét veszti, és a játék véget ér. Kiegészítésként még határozzunk meg annyit, hogy a játék kezdetét egy forgó, majd lefelé eső START GAME felirat jelzi, míg a végét egy GAME OVER felirat és egy YOU WIN felirat, amelyek forgás után újraindítják a játékot. Természetesen a ***Start*** felirat nem fogja újraindítani a játékot.
A fentiek alapján már tudjuk, hogy milyen tárgyakra lesz szükségünk. Készítsünk egy kockát, amit a hosszanti tengelye mentén méretezzünk át. Ez lesz majd a kalapács. Készítsünk egy újabb kockát, ezt azonban hagyjunk meg eredeti méretében, hogy a játék során megkülönböztethessük a kalapácsoktól. Ez a tárgy lesz – a sokszorosítás után – az álló akadály. Készítsünk két síklapot: az egyik lesz a pálya padlója, amin a figura mozoghat, a másik pedig azért fontos, hogy érzékelhessük azt, amikor a figura elhagyja a padlót és lefelé esve összeütközik ezzel a másik síklappal. Ezt a könnyebb elképzelhetőség érdekében tengernek vagy lávának nevezhetjük, így biztosan tudjuk, hogy káros lesz a játékfigura egészségére.
Szükségünk lesz még egy játékfigurára is. Az egyszerűség kedvéért most készítsünk egy gömböt, aminek az egyik vízszintes tengelye mentén húzzuk ki néhány pontját. Ez a pár kiálló pont lesz majd a figura „orra”, ebből tudja majd a játékos meghatározni a haladási irányát.

Az utolsó elkészítendő tárgy a kijáratot jelképezi. Ennél az olvasó képzelőerejére bízom a formai dolgokat, elegendő akár egy újabb kocka vagy gömb is, de lehet egészen bonyolult formát is alkotni. Az én példámban egy boltíves kaput készítettem. Elkészítettük tehát a tervet, megalkottuk a szereplőket, most már csak a működést kell vázolni, mielőtt a játék logikáját a Blender játékmotorjának a segítségével megvalósítanánk. Nagy vonalakban arról van szó, hogy amíg valamilyen felirat látható a képernyőn, addig a figura nem irányítható a billentyűzettel, és a tárgyak sem mozoghatnak, tehát az összes mozgó tárgynak a játékmotorban egy változóra lesz szüksége, amiből tudomást szerezhet a játék állapotáról. Amikor azonban semmilyen felirat nem szerepel, akkor a játék figyeli a billentyűzetleütéseket, így mozgathatjuk a figurát. Három tárggyal ütközhetünk. A padló alatti „tengerrel”, a „kalapácsokkal” és a kijáratot jelképező kapuval. Amikor a figura a kapuval ütközik, megjelenik a YOU WIN felirat, majd elindul egy számláló, amíg a felirat felfelé mozog, majd egyhelyben megáll. A meghatározott számérték elérése után a játék elölről kezdődik. Amikor a figura egy „kalapáccsal” vagy a „tengerrel” ütközik, szinte ugyanez a folyamat játszódik le, csak a GAME OVER felirat jelenik meg. A játék kezdetén megjelenik a START GAME felirat, meghatározott ideig forog, majd a másik számláló elindításával lefelé mozog. A figura és a kalapácsok mozgása a játék állapotát jelző változóhoz van kötve, csak annak igaz értéke mellett kell figyelembe venni.
Azt is tudjuk, hogy a kalapácsoknak pattogniuk kell, ez azonban az előző részek figyelmes elolvasása után már nem okozhat gondot. Tehát hozzuk létre a tárgyat, adjunk meg dinamikus anyagtulajdonságokat, az ütközés esetére pedig egy mozgatóerőt, amellyel az eredeti magasságába pattanhat vissza. Hogy ne kelljen minden ilyen veszélyes tárgyat újból létrehoznunk, a SHIFT-D billentyűkkel készítsünk róla másolatokat. A másolatokat mozgassuk el a helyükről, majd az utoljára létrehozottnak adjunk más tulajdonságokat, és erről szintén készítsünk másolatokat. Erre a lépésre azért volt szükség, hogy a játékos dolga ne legyen olyan egyszerű: más tulajdonságok esetén ugyanis más sebességgel fog pattogni az akadály.

```
Végtelen ciklus
Ha (aktív és ütközik(padló)) akkor
      erőhatás_felfelé
Ha (üzenet = "ok") akkor
      aktív := hamis
Ciklus vége
```

Miután minden mozgó akadályt elkészítettünk, helyezzük el őket a pályán. A mozgásukat vezérlő logika az *1. képen* látható. A jobb érthetőség kedvéért az *1. listán* egyszerű nyelven megfogalmazva láthatjuk a vezérlés logikáját.
Most következzen a játékfigura vezérlésének a megvalósítása. Szükségünk lesz egy logikai típusú tulajdonságra, amit



1. kép A kalapács logikája

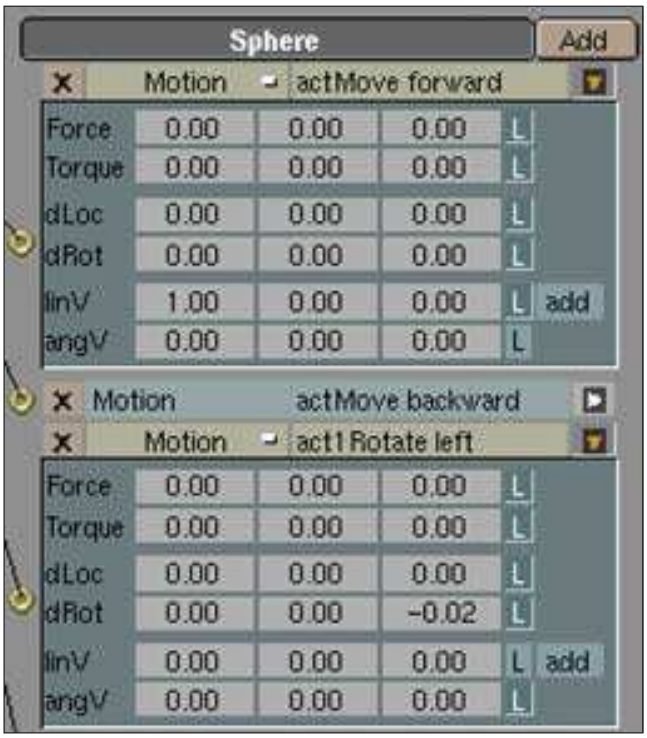
nevezzünk ***active***-nek. Ennek igaz értéke jelzi azt, hogy a játék fut és a figura irányítható. Továbbá a tulajdonságot lekérdező ***Propety*** érzékelőre is szükségünk van. Ezek mellett természetesen négy billentyűlenyomás-érzékelőt is létre kell hoznunk, mindegyik kurzormozgató gombhoz egyet-egyet. Ezeket a ***Keyboard*** típust választva készíthetjük el. Ne felejtsük el a megfelelő billentyűket beállítani. Az ütközések kezeléséhez készítsünk még három ütközésérzékelőt is (***Collision***), ezeknél anyagtulajdonságként a kapu, a kalapácsok és a „tenger” anyagát adjuk meg. A mozgás megvalósításához természetesen a mozgás tulajdonságaira ható ***Motion*** hatást kell létrehoznunk, szám szerint négyet. A gyorsításhoz és a lassításhoz (amelyeket a fel és le nyilakhoz rendelünk hozzá) egyet-egyet; ezeknek a `LinV` értéke fontos, itt adjunk meg 1-et, valamint -1-et az első koordináta szerint. A jobbra és balra nyilakhoz a Z tengely körüli elfordulást kell meghatároznunk, tetszés szerinti értékkel, de ne felejtsük el bekapcsolni a `Loc` kapcsolókat. Ezek a beállítások a *2. képen* láthatók.

Tudjuk, hogy a tárgynak csak akkor szabad mozognia, ha a játék futása megengedett, vagyis ha az ***active*** tulajdonság igaz. Valahogyan ezt a tulajdonságot is be kell állítanunk, tehát a kezdőértékét állítsuk hamisra, és hozzunk létre egy új érzékelőt, ***Message*** típussal. A ***Subj:*** mezőben adjuk meg a `start` szócskát, mert a későbbiekben ez az üzenet fogja jelezni a játék indulását. Az üzenetet majd a START GAME szöveg küli a jelenet minden aktív szereplőjének. Ehhez az üzenethez egy tulajdonságbeállító hatásnak kell tartoznia, ami az ***active*** értékét igazra fogja állítani. Most következhet az ütközések kezelése. Amikor a figura a kalapáccsal vagy a „tenger”-rel ütközik, egy üzenetet kell küldenünk a játék végét jelző szövegnek. Ez az üzenet legyen mondjuk „gover”, vagyis hozzunk létre egy üzenetet (***Message***) és a ***Subj:*** mezőbe írjuk be a „gover” szót.

Hasonló módon oldjuk meg a kapuval való ütközést is. Annyi különbség azért van, hogy az üzenetet nem a kapunak küldjük, hanem a YOU WIN feliratnak, és értelemszerűen más lesz az üzenet tárgysorában szereplő szöveg is.

Inudhat a játék

A létrehozott elemeket még össze kell kötni egymással. A billentyűzeteseményeket `AND` kapcsolatba kell hozni az ***active*** tulajdonságot érzékelő elemmel, így biztosíthatjuk, hogy csak a megfelelő időben tudjuk mozgatni a figurát. A kezdetet jelző érzékelő kimenetét egy `OR` logikai kapcsolaton keresztül kapcsoljuk a tulajdonság beállításához. Az ütközéseket csak a játék futása során kell figyelembe vennünk. Itt `OR` logikai kapcsolatot



2. kép Gyorsítás és fordulás

használhatunk. Következhet tehát a kezdőüzenet megvalósítása. Szükségünk lesz egy ***Timer*** tulajdonságra, ennek a kezdőértékét állítsuk nullára. Két értékvizsgálatot kell elvégeznünk a vezérlés során: az egyik szerint, amikor az időzítő értéke 1 és 1,45 között helyezkedik el, a szöveget a kamera síkjával párhuzamos tengely mentén egy kicsit el kell fordítanunk. Itt szándékosan nem adtam meg, hogy mennyi legyen az a „kicsi”, célszerű kikísérletezni, azt is figyelembe véve, hogy hányszor szeretnénk körbefordítani a szöveget, mielőtt lefelé mozogva kúszna a kamera látóteréből. Tehát egy ***Property*** érzékelőnek a típusát állítsuk ***Interval***-ra, és adjunk meg két értéket. Ehhez egy `OR` kapcsolattal kapcsoljunk hozzá

P	A játék indítása
ESC	A játék leállítása
Shift-F11	Parancsállomány-szerkesztő

```
Idő:=100
Végtelen ciklus
Ha ( Idő>1 és Idő<20) akkor
      szöveg_mozog_fel
Ha ( Idő > 40 ) akkor
      játék_újraindul
Növel( Idő )
Ha ( üzenet("win") ) akkor
      Idő:=0
Ciklus vége
```

egy ***Motion*** hatást, így a megfelelő értékek beállítása után a szöveg már képes lesz a kezdeti forgásra. A lefelé haladáshoz szintén az értéktartományt kell vizsgálnunk; hozzunk létre egy másik ugyanilyen érzékelőt, majd kapcsoljuk hozzá egy újabb ***Motion*** hatáshoz és egy ***Message*** hatáshoz is. A mozgás mértékét szintén tapasztalatunk alapján tudjuk majd megállapítani, de ne felejtsük el ennek a második értékvizsgálatnak az alsó határát nagyobbra állítani, mint amekkora az elsőként beállítottnak a felső határa volt. Amennyiben ezt elmulasztjuk megtenni, a szöveg a forgással egyidőben fog mozogni, ezt viszont most nem szeretnénk. Az üzenet tárgya legyen „start”, ezzel jelezzük az aktív tevékenységre képes tárgyaknak, hogy a játék elkezdődött.

Az eddigieket figyelemmel kísérve odáig jutottunk el, hogy a játék kezdetén van egy forgó, majd leeső START GAME felirat, s ezután irányíthatjuk a figurát. A kalapácsok fel-le mozognak, és amikor a figura alájuk kerül vagy leesik a pályáról, üzenetet küldünk valamelyik másik tárgynak. Amikor a figura a kapunak ütközik, a játékmotor felhasználásával szintén üzenetet küldünk. Mint látjuk, már nem maradt sok dolgunk. Az ütközésekkor keletkező üzeneteket a megfelelő helyen fel kell dolgozni.

A „win” üzenetet a YOU WIN feliratnak kell értelmeznie. Az üzenet hatására fel kell bukkannia, majd rövid várakozás után újra kell indítania a játékot. A felbukkanást könnyen megoldhatjuk úgy, hogy a játék kezdetén a szöveget a pálya alatt helyezzük el, és az üzenet hatására egy időzítő értékét változtatjuk meg. Ezután, amíg az időzítő egy bizonyos tartományon belül tartózkodik, a szöveg felfelé mozog; majd egy másik értéktartományba belépve a játék elölről kezdődik. A dolgok gyakorlati oldalát nézve: hozzunk létre egy ***Timer*** típusú tulajdonságot a szöveg számára. Ennek a kezdőértékét állítsuk mondjuk 100-ra, így elkerülhetjük, hogy már a játék kezdetekor felvegye azokat az értékeket, amikre csak később lesz szükség. Hozzunk létre két értéktartomány-érzékelőt ***Property – Interval***, és a ***Prop*** mezőben adjuk meg az időzítő azonosítóját. Az első érzékelési tartománya legyen például 1–20, ez fogja vezérelni a mozgást, és egy OR kapcsolaton keresztül kapcsoljunk is hozzá egy mozgásra vonatkozó hatáselemet. A mozgás jellemzői ugyanazok lehetnek, mint az előző felirat esetében.

A másik érzékelő alsó határát állítsuk negyvenre, a felső határát pedig 99-re. Talán érthető, hogyha itt a felső határt 100-nál nagyobbra állítottuk volna, már a játék kezdetén véget érne a működése, mert az időzítő kezdőértékét 100-ra állítottuk. Az első vizsgálat felső határa és a második vizsgálat alsó határa közötti különbség (esetünkben ez az érték 20) adja meg azt az időt, amíg a felirat nem forog, és még a játék sem indul újra. Itt figyelni kell arra, hogy a forgás befejezésekor a felirat a megfelelő elfordulással jelenjen meg.

A játék újraindításához egy ***Scene*** típusú hatás kell az érzékelőhöz kapcsolni, majd ki kell választani a legördülő listából a ***Restart*** tételt. Most már minden rendben is volna, ha a felirat valamilyen módon tudomást szerezne arról, hogy mikor is kellene elkezdenie a mozgást.

A megoldás az lesz, hogy egy ***Message*** érzékelőnek beállítjuk a tárgy mezőjében a „win” szócskát (amit a figura–kapu-ütközéskor majd a játékfigura küld a jelenetben résztvevőknek), és egy OR kapcsolat segítségével egy olyan hatást kapcsolunk hozzá, ami valamilyen tulajdonságot állít be. Ennek a tárgynak az egyetlen tulajdonsága az időzítő értéke, vagyis ezt nullára kell beállítanunk. Ezzel elérjük, hogy a kapuhoz érve az időzítő nulla értéket kap, majd folytatja természetes működését, vagyis az értéke növekedni kezd. Egy és húsz közötti értékek között a szöveg felfelé mozog, ezután 20–40-ig nem történik semmi. A negyvenes értéket elérve a játék a ***Scene*** hatás működése miatt elölről kezdődik. A fentiek megértését segíti a 2. *lista*.

A játékos megsemmisülésekor ugyanezt a vezérlést kell megvalósítani, csak ebben az esetben a GAME OVER felirathoz kell hozzárendelni a megfelelő elemeket, és az üzenet tárgysorában a „win” szöveg helyett a „gover” szövegnek kell szerepelnie, hiszen – mint korábban már beállítottuk – a kalapáccsal vagy a „tenger”-rel ütköző figura ezt az üzenetet küldi a jelenet résztvevőinek.

Nos, ennyi lenne egy egyszerű játék elkészítése a Blender játékmotorjának a felhasználásával. A játékot a P billentyűvel indíthatjuk el, és máris észrevehetjük eddigi munkánk első szépséghibáját. Látható, hogy minden fehér színben és árnyékolás nélkül jelenik meg. Ha a megjelenítést a D billentyűvel előhívható menüből választva árnyékoltra változtatjuk, és így indítjuk el a játékot, látjuk, hogy ettől sem lett jobb a helyzet. Nos, ha figyelemmel kísérték a sorozat korábbi részeit, nem fog gondot okozni a felületi mintázatok elkészítése. Szükség is lesz e munka elvégzésére, ugyanis a Blender csak ilyen mintázatokkal ellátott tárgyakat jelenít meg látványosan a játékmotor elindítása után. Ezt a lépést sajnos nem célszerű a teljes vezérlés és a környezet kialakítása után elvégezni, mert így elveszítjük azt a lehetőséget, hogy a másolatként keletkezett tárgyaknak már megvan a mintázata és minden egyéb tulajdonsága is. Ha most szeretnénk elkészíteni a mintázatokat, akkor azokat minden tárgyon külön kellene beállítanunk. Ez véleményem szerint felesleges lenne, létezik más megoldás is. Ahhoz, hogy a mostani munkánk se vesszen kárba, nem kell teljesen elölről kezdeni mindent, elég, ha minden másolatból csak egyet hagyunk (egy kalapácsot és egy álló akadályt), majd ezeket mintázatokkal ellátva megismételjük a másolatok készítését és elhelyezését.

A mintázatok

Természetesen a játék még nincs befejezve, de a célom nem is ez volt, hanem az, hogy bemutassam egy jól használható modellező és animációs program képességeit, olyan eszközöket és módszereket adva az érdeklődők kezébe, amelyek nem csupán szórakoztató és unaloműző tevékenységre alkalmasak, hanem az egyéni önkifejezésre is.

A továbbiakra nézve sok-sok jó ötletet és az alkotás örömét kívánva búcsúzom.

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

Héjprogramozás: átmeneti fájlok és jelek (8. rész)

Sorozatunk mostani részében folytatjuk azoknak az általános programozási módszereknek és eljárásoknak a bemutatását, amelyekkel gyakran találkozhatunk munkánk során.

Egy program megvalósítása során olyan helyzetbe is kerülhetünk, amikor a feldolgozás valamiféle „köztes termékét” átmenetileg kénytelenek vagyunk egy vagy több fájlban tárolni. Például ilyen helyzet akkor állhat elő, amikor egy bizonyos művelet végrehajtásához a bemenet egészéről kell valamilyen adattal rendelkeznünk. Ha az adatokat fájlból vesszük, nincs más dolgunk, mint kétszer végigolvastatni a tartalmukat. Az első olvasás alapján meghatározhatjuk a kívánt globális jellemzőket, a másodikkal pedig elvégezzük a feldolgozást. Átmeneti tárolásra, illetve fájlra ebben az esetben nincs szükségünk, hiszen a bemenet eleve „tárolt formában” áll rendelkezésünkre. Ez így eléggé elvontan hangzik, ezért lássunk rögtön két példát!

Tegyük fel, hogy egy szövegfájl tartalmát bekeretezve akarjuk megjeleníteni a képernyőn. A keretnek nyilván igazodnia kell a szöveg szélességéhez, vagyis a leghosszabb sor hosszához. Azt viszont, hogy milyen széles a leghosszabb sor, csak úgy tudjuk megállapítani, ha az egész fájlt végigolvassuk.

Hasonló a helyzet, ha a szöveg sorait mondjuk fordított sorrendben szeretnénk kiíratni. Előbb tudnunk kell, hogy melyik (hányadik) az utolsó sor, csak azután kezdhetünk el visszafelé haladni a kiíratással.

Ha fájlból vesszük a szöveget, mindennek semmi akadálya. Mi a helyzet azonban akkor, ha a unixos szokásoknak megfelelően programunkat szabványos bemenetről érkező szöveg feldolgozására is képessé akarjuk tenni? Honnan tudhatjuk egy szöveg elején, hogy hány karakterből fog állni a leghosszabb sora, vagy hogy hány sorból fog állni a teljes bemenet? Természetesen sehonnan. Ezeknek a kiderítéséhez legalább egyszer végig kell olvasnunk a teljes bemenetet, ami jelen esetben azért gond, mert a szabványos bemenet csak egyszer olvasható.

Ha a bejövő csőből kiolvastunk egy sort, azzal ki is töröltük onnan, vagyis már nem lesz mit feldolgozni. Mi tehát a megoldás? Természetesen az, hogy a bemenetet – akár az első olvasással párhuzamosan – egy átmeneti fájlba irányítjuk, és a következő lépésben az így „bespájzolt” adatot dolgozzuk fel.

A dolog miként je

A következő feladat az átmeneti fájlok elhelyezése. Megtehetjük ugyan, hogy egy adott néven mindig a pillanatnyi könyvtárban vagy mindig egy adott könyvtárban helyezzük el programunk átmeneti fájljait, de ebből mindenféle galibák származhatnak. Nem biztos például, hogy mindig írási jogunk van a pillanatnyi könyvtárhoz. Ha van, akkor viszont abban nem lehetünk biztosak, hogy nem futtatja a programot velünk egy időben más is. (Ne felejtsük el, hogy a unixok egyik fő erőssége a többfeladatos, többfelhasználós működés.) Márpedig ha egyszerre többen használják ugyanazt az átmeneti fájl, abból minden kétséget kizáróan irgalmatlan keveredés származik majd. Gondoskodnunk kell tehát valahogyan az átmeneti fájlok egyediségéről, illetve használat utáni eltakarításukról is.

A Unix-rendszereken és így a Linuxon is éppen ezt a célt szolgálja a */tmp* könyvtár, illetve a folyamatazonosító. A */tmp* amolyan közös szemétdomb, ahová mindenki írhat, de az „általános írási jogosultság” ellenére egy különleges rendszerszolgáltatásnak, az úgynevezett ragasztó bitnek (sticky bit) köszönhetően a fájlokat kizárólag a tulajdonosuk törölheti. A folyamatazonosító minden egyes futó programpéldányra egyedi, ezért ha ezt szerepeltetjük az átmeneti fájlok nevében, automatikusan kizárt a keveredés lehetősége. Héjprogramokban

```
 1: #!/bin/sh
 2:
 3: vissza()
 4: # A $1-ben megadott fájl sorait olvassa
      ➥visszafelé
 5: {
 6:   hossz=`cat $1 | wc -l`
 7:   while [ $hossz -ge 1 ]
 8:   do
 9:     cat $1 | sed -n "$hossz p"
10:     hossz=`expr $hossz - 1`
11:   done
12: }
13:
14: # Fájlokból érkező bemenet
15: if [ $# -ne 0 ]
16: then
17:   for nev in $*
18:   do
19:     if [ -f $nev ]
20:     then
21:       vissza $nev
22:     else
23:       echo "A(z) $nev nevű fájl nem
                ➥létezik!" 1>&2
24:     fi
25:   done
26: else  # A szabványos bemenet olvasása
27:   touch /tmp/tactmp$$
28:   trap 'rm /tmp/tactmp$$ ; exit 1' 1 2 3
15
29:   while read sor
30:   do
31:     echo $sor >> /tmp/tactmp$$
32:   done
33:   vissza /tmp/tactmp$$
34:   rm /tmp/tactmp$$
35: fi
36: exit 0
```

a folyamatazonosítót a `$$` belső változó tartalmazza, így például a `/tmp/tmp$$` *név* minden korábban említett igényt kielégít.

A héjprogramokban kezelhető fontosabb jelek

Kiváltó esemény	A jel neve	A jel száma	Leírás
exit parancs	**EXIT**	0	**exit** parancsot hajtott végre a program vagy normál úton befejeződött.
kijelentkezés	**SIGHUP**	1	Megszakadt a kapcsolat a terminállal.
Ctrl+C	**SIGINT**	2	A felhasználó leállította a programot
Ctrl+\	**SIGQUIT**	3	Kilépésre való utasítás a billentyűzetről.
kill -9	**SIGKILL**	9	Feltétel nélküli leállás. (Nem fogható el!)
shutdown vagy **kill**	**SIGTERM**	15	Felszólítás leállásra.

Egy példa

Példaként az előbb említett két feladat közül valósítsuk meg a fájl sorainak fordított sorrendben való megjelenítését. (Ezt a célt szolgálja egyébként a legtöbb rendszeren megtalálható `tac` parancs is, így erre a feladatra héjprogramot írni inkább csak amolyan ujjgyakorlat.) Mint megannyi más esetben, a tényleges feldolgozást végző kódot most is függvény formájában célszerű megvalósítani, amely a feldolgozandó fájl nevét kapcsolóként kaphatja meg (lásd a *listában*).

A program lelke a 7–11. sorban látható. A 6. sorban megszámoltuk a bemenet sorait, tehát már tudjuk, hogy melyik az utolsó. Ezután egy `while` ciklusban ezt a mutatót lépésenként csökkentjük (10. sor), és minden egyes fordulónál az adott sorszámú sort kiíratjuk – az utóbbihoz a `sed "p"` parancsát használjuk. Természetesen ugyanezt `awk`-val vagy egy `head-tail` párossal is megoldhattuk volna, de talán ez a legegyszerűbb.

A program maradéka a vezérlést ellátó főprogram. Azt, hogy fájlokból kell vennie a bemenetet, onnan ismeri fel a program, hogy a parancssori kapcsolók száma nem nulla (15. sor). Ilyenkor egy `for` ciklussal (17. sor) valamennyi néven végigmegy, és egyenként átadja őket a feldolgozást végző függvénynek. Amennyiben valamelyik fájl neve hibás, hibaüzenetet kapunk.

A számunkra igazán érdekes rész a 26. sorban kezdődik.

A 27. sorban létrehozunk egy – pillanatnyilag teljesen üres – átmeneti fájlt, amelynek a neve a `tactmp` karakterláncból és a héjprogram folyamatazonosítójából (`$$`) áll össze. A 28. sorral egyelőre ne foglalkozzunk.

A bemenetről érkező szöveget a 29–32. sorban látható `while` ciklus ebben az átmeneti fájlban egyszerűen összegyűjti , majd ha minden készen áll, a program ezúttal az átmeneti fájl nevével hívja meg a feldolgozó függvényt (33. sor).

Takarítás

Egy „rendes” program természetesen arra is fel van készítve, hogy az általa létrehozott „kacatokat” eltakarítsa maga után, még mielőtt kilépne. Mi sem egyszerűbb ennél, gondolhatnánk elsőre: ha megtörtént a feldolgozás, egyszerűen töröljük az átmeneti fájlt, és készen is vagyunk. Jelen esetben a 33. sor után kell kiadnunk az `rm /tmp/tactmp$$` parancsot. Látható azonban, hogy ez a törlés valamilyen furfangos módon már a 28. sorban is szerepel egy `trap` parancs részeként.

Ezt a megoldást az indokolja, hogy általában nem lehetünk biztosak benne, hogy a program végrehajtása valóban eljut a 33. sor utáni részig. A legegyszerűbb esetben előfordulhat például, hogy a felhasználó a feldolgozás kellős közepén megnyomja a Ctrl+C billentyűket, és megállítja a programot. Ilyenkor az átmeneti fájl minden igyekezetünk ellenére megmarad, hiszen a programnak esélye sincs rá, hogy törölje. Hasonló a helyzet, ha a futó programot a `kill` paranccsal „kilőjük”, vagy a rendszergazda – esetleg az áramszolgáltató és a szünetmentes tápegység – kifürkészhetetlen kegyéből a program végrehajtása során rendszerleállás következik be.

Végzetszerűségük ellenére az efféle események nem teljesen váratlanul következnek be. A program előbb kap egy értesítést, egy úgynevezett jelet a rendszermagtól, és valójában erre való válaszul fejezi be a működését. Mármost, ha van előjele a „katasztrófának”, akkor utolsó intézkedésként tehetünk még valamit, amivel menthetjük a menthetőt – például törölhetjük az átmeneti fájlt.

A jelek kezelése

A jelek elfogására és kezelésére szolgál a `trap` parancs, amelynek általános alakja a következő:

```
trap 'parancsok' jelek neve vagy száma
```

A `trap` a felsorolásban megadott jeleket elfogja, és hatásukra az egyszeres idézőjelek (aposztrófok) között megadott utasítássort hajtja végre. A legfontosabb jelek számát, nevét és leírását a mellékelt táblázat tartalmazza.

A `trap` csak attól a ponttól kezdve érvényes a programban, ahol ő maga szerepel. Ha tehát valamilyen globális, a program egészére vonatkozó műveletet akarunk végrehajtani egy `trap` részeként, azt célszerű rögtön a kód elején megadni. Ha viszont csak egy adott blokkban akarunk jelkezelést megvalósítani (esetünkben ez a helyzet), akkor a kérdéses műveleteket elegendő ennek a blokknak az elején szerepeltetni. A blokkból kilépve a különleges jelkezelést olyan módon meg is szüntethetjük, hogy egy műveletek nélküli `trap` parancsot adunk ki az adott jel számával vagy nevével.

Programunkban az egyetlen olyan művelet, amit „végszükség esetén” is végre akarunk hajtani: az átmeneti fájl törlése. Ennek megfelelően a 28. sorban látható `trap` parancs valamennyi, a program leállásához vezető jelet elfogja, végrehajtja a törlést, majd az 1-es visszatérési értékkel kilép. Az utóbbira azért van szükség, mert egy jel kezelése nem jelenti önműködően a program leállását. Ha tehát a `trap` részeként nem hajtunk végre kilépést, akkor a program tovább fut!

Azt is érdemes megfigyelni, hogy a 28. sorban a jelek felsorolása nem tartalmazza a 0-s jelet, vagyis a program szokványos leállását. Erre azért van szükség, mert ha a program a szokásos módon áll le, vagyis egyszerűen vége szakad, akkor előtte biztosan törölte az átmeneti fájlt. Így viszont a `trap`-ben szereplő `rm` parancs hibát jelezne.

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

Hálózatok (1. rész)

Magazinunk lelkes és kitartó olvasói az elmúlt több mint egy évben végigizgulhatták az operációs rendszerek belsejét feltáró sorozatunkat. Ez azonban véget ért, de nem maradunk érdekfeszítő téma nélkül. Következzenek most a hálózatok!

Az operációs rendszereket tárgyaló sorozatban tudatosan kerültünk minden olyan témát, amelynek bármilyen köze is lehetne a hálózatokhoz. Ennek az az oka, hogy egy életképes hálózati modell működése csaknem olyan bonyolult, mint egy operációs rendszeré. Ezért új sorozatunk hasonló lehetetlen feladatra próbálkozik: megpróbálja bemutatni, hogyan is működik egy világméretű hálózat, például az internet.

Aki már nagyon unja az operációsrendszer-témát, és épp örömtüzeket gyújtott, hogy végre teljesen más vizekre evezünk, az sajnos téved. A számítógépes hálózatok szorosan összefüggenek az operációs rendszerekkel. A legfontosabb hálózati szolgáltatások megvalósításáért ugyanis a legtöbb esetben maga az operációs rendszer a felelős. Gondoljunk a Linux-rendszermagra, amely az adatátviteli protokollok (TCP, UDP) megvalósítása mellett még egy csomagszűrő tűzfalat is tartalmaz. De létezik egy olyan operációs rendszer is, amelynek a fejlesztői (valószínűleg inkább üzleti, mint szakmai megfontolásból) úgy gondolják, jó dolog, ha a böngésző is a rendszer egyik szerves, éppen ezért elválaszthatatlan részét képezi.

Ez a sorozat tehát némileg az előzőre épül, de remélhetőleg azoknak is minden érthető lesz, akik az operációs rendszerekről szóló elmélkedéseinket nem olvasták. Hogy pontosan miről lesz szó az elkövetkezendőkben, nehéz pár mondatban leírni. Mindenképpen az alapdolgokkal kezdünk, aztán rátérünk a hálózati eszközökre, majd a hálózati programokra. Az elején mindennek az elméleti oldalával ismerkedünk meg, majd a gyakorlatban is bemutatjuk: megvizsgáljuk, miként is működik maga az internet. Mindezek mellett természetesen linuxos „vonatkozása” is lesz sorozatunknak, azaz a gyakorlati példákat a Linux-rendszermag hálózati részén mutatjuk be, és természetesen szó lesz arról is, hogy mire jó az a sok szolgáltatás, amit belefordíthatunk a rendszermagunkba.

Úgy illik, hogy először megmondjuk, mi az a számítógép-hálózat – olvasóink többsége azonban biztos sértésnek venné, ha most több oldalon keresztül elmagyaráznánk, miről is szól ez a sorozat. Így hát rövidre fogjuk: ma már a számítógép olcsó, simán vehetünk egyet minden munkatársnak az irodába. Ha egy csomó gépünk van, felmerül az igény, hogy összekössük őket, egyrészt azért, hogy az adatokat könnyebben átvihessük egyik gépről a másikra, másrészt azért, hogy ne minden géphez egyenként kössük be az internetet. Megállapíthatjuk tehát: a számítógépes hálózat hétköznapi jelenség.

Ez a jelenség azonban nemcsak mindennapi, de viszonylag új is. Volt idő, amikor a számítógép drága volt, és nem alkalmazottanként, hanem cégenként akadt egy belőle (már ha az adott cégnek olyan jól ment, hogy beruházhatott egy ilyen masinába). Ezt az egy gépet használta minden felhasználó, akik terminálok segítségével dolgoztak.

Az első számítógépes hálózatok története arra az időre nyúlik vissza, amikor a gépek ára annyira leesett, hogy a cégek akár két masinát is tudtak venni. Ekkor kezdtek elgondolkozni azon, hogy milyen jó lenne, ha meg lehetne oldani, hogy a különálló gépek összekapcsolódhassanak egymással. Ez volt a hálózatok megszületésének pillanata – a további történéseket pedig már ismerjük.

A fentiekből adódóan nem tekinthetjük hálózatnak az olyan rendszereket, ahol egy központi géphez olyan más gépek kapcsolódnak, amelyek között egyértelműen mester–szolga viszony alakult ki, ilyenek például a terminálok.

Most jöjjön egy nehéz kérdés: hálózat-e vajon egy osztott rendszer, másnéven telep? Természetesen nem. A telepben ugyan fizikailag különálló gépek vannak összekötve, de a felhasználó nem látja külön-külön a gépeket, az egész hálózat egyetlen számítógépként viselkedik. Amikor a felhasználó létrehoz egy állományt, nem tudja, hogy melyik gép merevlemezén lesz tárolva. Amikor a felhasználó elindít egy programot, nem tudja, hogy az melyik gép processzorán fog futni – ezeket a dolgokat az operációs rendszer dönti majd el.

Egy hálózatban azonban a felhasználó pontosan meg tudja nevezni, hogy éppen melyik gép erőforrását használja.

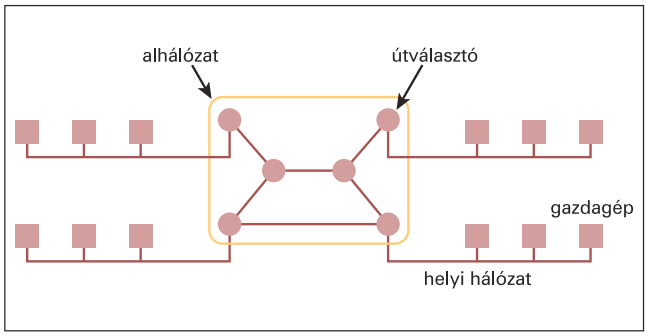
(A telepek esete valójában bonyolultabb. A gépek tulajdonképpen hálózatba vannak kötve, és az adatok továbbítására is az elkövetkezendőkben bemutatott módszerek valamelyikét használják. A különbség tehát nem a vasban, hanem inkább a hálózati programban van: a telep nem más, mint egy olyan programrendszer, amely a már meglévő és működő hálózatra ráépülve a gépeket elrejti a felhasználók szeme elől, azt az érzetet keltve bennük, hogy csak egyetlenegy gép van, amely jóval nagyobb kapacitással bír, mint a hálózatba kötött gépek bármelyike).

A hálózati eszközök

Ha csoportosítani szeretnénk az összes színes és szagos hálózatot, amely kis világunkban fellelhető, egyből komoly nehézségeink adódnának. A nehézségeket az a tény okozná, hogy nem létezik egy olyan általános osztályozás, amelynek alapján mindenfajta hálózatot beskatulyázhatnánk.

Azért nincs veszve minden, létezik ugyanis két olyan tényező, amely alkalmas kiindulási pontot jelenthet egy-egy hálózat vizsgálgatásakor. Az első ilyen tényező az úgynevezett átviteli technológia.

Kétféle átviteli technológia létezik: az üzenetszórásos és a kétpontos. Kezdjük az üzenetszóróssal: ebben az esetben a hálózatban jelenlévő gépek egy közös csatornán osztoznak. Ha valaki beleszól ebbe a csatornába, azt mindenki meghallja. Ha az egyik gép üzenetet szeretne küldeni a másik gépnek, akkor először meg kell mondania, hogy kinek küldi, majd el kell mondania az üzenetet. A többi gépnek ezért minden üzenetet meg kell vizsgálnia, hogy neki szól-e. Ha ő a címzett, akkor a bejövő üzenetet fel kell dolgoznia, ha nem, akkor figyelmen kívül hagyja. Ez a dolog nagyon hasonlít a CB-rádiók

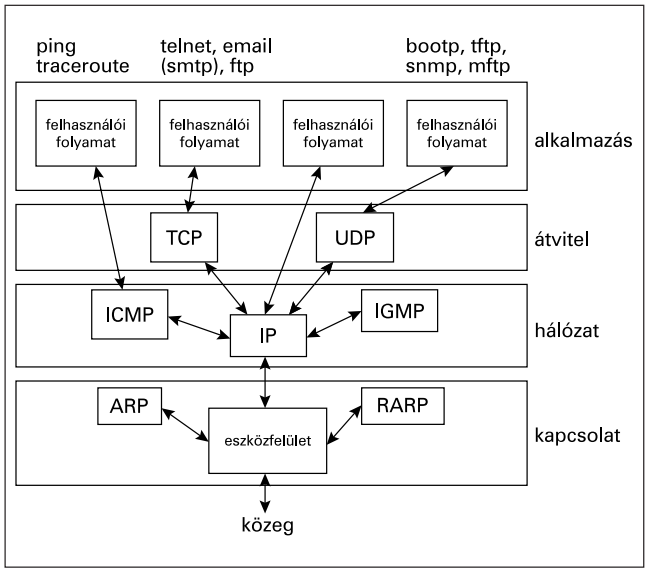


1. ábra
A gazdagépeket egymással az útválasztók hálózata, más néven az alhálózat köti össze. Egy útválasztóhoz nemcsak egy gazdagép, hanem egy egész gazdagépekből álló helyi hálózat is kapcsolódhat

működéséhez, de aki utazott már repülővel, az is részesévé vált egy üzenetszórásos kapcsolattartásnak. Gondoljunk csak arra, amikor a hangosbemondó elmondja, hogy melyik járat utasai melyik kapuhoz fáradjanak a beszálláshoz.
Az összes ezen az elven alapuló hálózatban létezik egy olyan szolgáltatás, amellyel egy gép üzenetet küldhet az összes másiknak – ez maga az üzenetszórás (broadcasting). A gyakorlatban ez úgy szokott működni, hogy létezik egy különleges cím, és az összes oda küldött üzenet az összes géphez eljut. Bizonyos rendszerek azt is lehetővé teszik, hogy a hálózat gépei csak egy bizonyos csoportjának küldjünk üzenetet; ez a csoportszórás (multicasting), megvalósítására pedig sokféle módszer kínálkozik.
Az ilyen elven működő hálózatok (mint például az ethernet-hálózatok) egy komoly biztonsági hiányosságot hordoznak magukban: a kapcsolattartás bárki számára könnyedén lehallgatható. Ez az úgynevezett szaglászás (sniffing), és a hallgatózók (snooping) ellen csak a kapcsolattartás titkosításával védekezhetünk.
Az üzenetszórás teljes ellentéte a kétpontos átviteli technológia. Itt a gépeket párosával kapcsoljuk össze, azaz egy csatornán csak két gép osztozhat. Mivel egy nagyobb hálózatnál nem oldható meg, hogy minden géphez egy, a hálózat összes többi gépéhez vezető külön csatorna tartozzon, az üzenetek általában több gépen is keresztülmennek, mire célba érnek. Az összekapcsolástól függően két gép között több út is létezhet, ezért megjelenik a forgalomirányítás gondja. Erről majd részletesen is szólunk, amikor az útvonalválasztók (router) kerülnek terítékre.

Hálózatok mérete

Akad még egy olyan jellemző, amivel minden hálózat rendelkezik. Bizony, most jönnek az informatikaórák kedvelt tananyagaként szereplő LAN (helyi hálózat), MAN (városi hálózat) és WAN (széles kiterjedésű) hálózatok. A különbség közöttük nem merül ki annyiban, hogy míg az első csak egy épületnyi, addig a második már egy városnyi és az utolsó gigászi méretű hálózat.
A LAN (Local Area Network, azaz helyi hálózat) legfontosabb ismérve az, hogy kis kiterjedésű, általában egy, esetleg két épület; ennél nagyobb azonban nem lehet. Mivel a mérete korlátozott, létezik egy jól behatárolható időtartam, ami alatt a küldött adat biztosan megérkezik.
A leggyakoribb helyi hálózat az úgynevezett sínkiépítés (topológia), amikor a gépek egyetlen kábelre vannak felfűzve. Itt egyértelműen üzenetszórást alkalmazunk. Az ilyen megoldásnál fontos, hogy egyszerre mindig csak egy gép adhasson, különben gubanc lesz. Ennek irányítására ezért különböző szabványokat vezettek be. Ezek közül messze legismertebb az ethernet, amely a nehézséget a következőképpen oldja fel: minden gép akkor ad, amikor csak jólesik neki. Ha azonban két vagy több csomag ütközik, akkor a küldeni kívánó gépek mind véletlenszerű ideig várakoznak; ezután ismét megpróbálkoznak a küldéssel.
A MAN (Metropolitan Area Network, vagyis nagyvárosi hálózat) már egy picit nagyobb terjedelmű dolog, mint a helyi hálózat, de igazából nem sokban különbözik. Azért érdemes mégis szólni róla pár szót, mert létezik egy szabvány, ami egyre jobban kezd elterjedni, ez pedig a DQDB (Distributed Queue Dual Bus). A lényege az, hogy húznak két egyirányú sínt, amelyre rácsatlakoztatják a számítógépeket. Mindkét sínhez egy főállomást rendelnek, ez felel az átvitel irányításáért. Ha egy gép a tőle jobbra lévő gépnek szeretne adatot küldeni, akkor a felső sínre, ha pedig a balra lévőnek, akkor az alsó sínre kerül az adat. Fontos még megjegyeznünk, hogy egy



2. ábra
A képen példát láthatunk arra, hogy milyen rétegekből is épülhet fel egy hálózat. Ezt a „felosztást" használja az internet is

nagyvárosi hálózat csakis akkor lehet hatékony, ha van olyan csatorna, amihez a gépek könnyedén kapcsolódni tudnak.
A WAN (Wide Area Network, széles kiterjedésű hálózat) hatalmas terjedelmű, országos, esetleg földrésznyi méretű hálózat. Kétféle gépet foglalhat magában: az egyik az a fajta, amelyik képes a felhasználói programok futtatására: ezeket gazdagépeknek (host) nevezzük. A gazdagépeket úgynevezett kapcsolattartási alhálózat köti össze a többi gazdagéppel. Az alhálózat többnyire átviteli vonalakból és kapcsolóelemekből áll. (Az elnevezés megtévesztő, mivel ez az alhálózat nem azonos azzal az alhálózattal, amellyel a hálózati címzésnél találkozhatunk.)
A kapcsolóelemek képezik a másik gépcsoportot, amely egy széles hálózatban lelhető fel. Míg a gazdagépek „egyszerű" számítógépek, amelyeken a felhasználók alkalmazásokat futtatnak, addig ezek a kapcsolóelemek különleges masinák, és semmi másra nem jók, mint hogy két vagy több átviteli vonalat összekapcsoljanak. Mi ezeket útválasztónak (router) nevezzük. Ezt megvilágítandó vessünk egy pillantást a *3. ábrára*. Ezen azt látjuk, hogy az egymáshoz közel lévő gazdagépek egy helyi

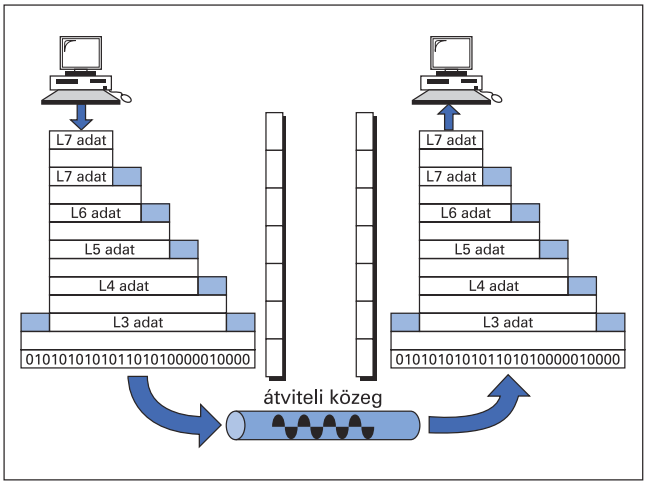
hálózatra vannak felfűzve, és ez a helyi hálózat kapcsolódik egy útválasztóhoz. Természetesen az is lehet, hogy egy útválasztóhoz csak egyetlen gazdagép kapcsolódik. Az ábrából az is kiderül, hogy az útválasztók összessége alkotja az alhálózatot, amelybe a gazdagépek nem tartoznak bele (ez nagyon fontos dolog!). Az ilyen nagy kiterjedésű hálózatoknál gyakori, hogy egy útválasztó egy olyan útválasztónak akar adatot továbbítani, amelyhez nem rendelkezik közvetlen kapcsolattal. Ilyenkor csakis közvetett módon, más útválasztók segítségével tudja célba juttatni az adatot.

Összekapcsolt hálózatok

Kétségtelenül számos előnnyel jár, ha számítógépeinket összekapcsolhatjuk és hálózatokat hozhatunk létre. Ennél már csak az lenne előnyösebb, ha a hálózatokat is összekapcsolhatnánk. Itt azonban felmerül egy komoly gond. Amikor hálózatokat építünk, akkor tudatosan úgy vásároljuk az eszközöket, hogy azok egymással minden téren együttműködők legyenek.
Ha azonban két független hálózatot szeretnénk összekapcsolni (amelyeket valószínűleg két különböző ember épített), akkor könnyen elképzelhető, hogy a két hálózat mind az eszközök, mind a pogramok terén különbözni fog.
Ezért egy másik különleges számítógépre van szükség, az átjáróra (gateway), amely – mint nevében is benne van – biztosítja az átjárhatóságot két hálózat között. Az átjárókkal összekapcsolt hálózatokat internetworknek, röviden csak internetnek nevezik.
Az internetre a legjobb példa maga az Internet, amely egy, az egész világot behálózó, összekapcsolt hálózat. Sőt ma már ennél sokkal több. Mára ugyanolyan közvetítőközeggé lett, mint a tévé meg a rádió, ezért igazából kisbetűvel illik írni, hiába tulajdonnév és csak egy van belőle, hiszen a világból is csak egy van, mégsem írjuk nagy kezdőbetűvel. Akárhogy is, ennek fejtegetése nem tartozik sorozatunk témájához, és a továbbiakban az (I/i)nternetre kizárólag szakmai szemmel tekintünk. Ezért kénytelenek vagyunk a következő kikötéseket tenni: amikor azt mondjuk, hogy internet, akkor általánosságban az összekapcsolt hálózatokról beszélünk. Amikor pedig az mondjuk, hogy Internet, erről a világméretű hálózatról lesz szó.

A szerkezet

Az odáig rendben van, hogy a gépeket kábelekkel, esetleg rádió-adóvevőkkel összedugdossuk. Ez azonban édeskevés, mivel ezzel csak a csatornát teremtettük meg, amin keresztül a kapcsolattartás zajlik – a gépeknek ugyanis szót is kell érteniük egymással. Ekkor jut szerephez a hálózati program.
Hogy a hálózati program ne legyen túl bonyolult, minden hálózat esetében az a bevált szokás, hogy rétegekre (ha úgy tetszik szintekre) bontják. Minden rétegre csak egyetlen feladat hárul, és az adott feladat megoldása teljesen független a többi rétegtől. A továbbiakban egy hétköznapi életből vett történet segítségével próbáljuk meg bemutatni, hogyan is működik a hálózati program. Számos alapvető, ám rendkívül fontos dologról lesz szó az elkövetkezendőkben, elképzelhető, hogy olvasóink többségének ezek ismerősek lesznek. Ugyanakkor lényeges, hogy átbeszéljük ezeket, mivel egész cikksorozatunk erre fog épülni.
Azért – a félreértések elkerülése végett – azt előrebocsátanánk, hogy a hálózati program nem olyan program, amit az ember boltban megvásárol CD-n, vagy letölti valamelyik warez-oldalról. A hálózati program tulajdonképpen egy programcsomag, amelynek a legalsó rétege az ethernetkártyánkba van beleégetve, a középső rétegek az operációs rendszer részei, és a legfelső szintű részeket pedig a felhasználó telepíti és használja (például amikor a Netscape-pel böngészik a neten). Ezeket így együttesen hálózati programoknak nevezzük.



3. ábra
Ez az ábra remekül szemlélteti, miként is jut el az adat az egyik gépen futó felhasználói alkalmazástól a másik gépen futó folyamathoz. Ahogy a csomag egyre alsóbb réteghez kerül, a mérete egyre nő, mivel minden réteg tesz hozzá valamilyen extraadatot. Fontos, hogy fizikai kapcsolat csak a legalsó rétegek között zajlik

A történet tehát a következőképp hangzik: él két filozófus, akik szakmai vitát szeretnének folytatni egymással az élet értelméről. A gond azonban abból adódik, hogy az első filozófus csak angolul, a másik meg csak németül tud. Ezért felfogadnak maguk mellé egy-egy tolmácsot. Az angolul tudó filozófus elküldi kutatásának az eredményeit a tolmácsának, aki a másik tolmáccsal már megállapodott egy semleges nyelv használatában (legyen ez a spanyol). A tolmács az üzenetet lefordítja spanyolra, majd átadja titkárnőjének, hogy küldje el a másik tolmácsnak. A titkárnő megegyezik a német filozófus tolmácsának titkárnőjével (reméljük, a történet még követhető), hogy elektronikus levélben küldi el az anyagot. Miután ez megtörtént, a német filozófus tolmácsa lefordítja a spanyol szöveget németre, majd átnyújtja a főnökének.
Ez a mese nem bővelkedett ugyan izgalmas fordulatokban, és csattanó sem volt a végén, mégis számos tanulsággal szolgálhat. Figyeljük meg, hogy a filozófusokat egyáltalán nem érdekli, hogy a két tolmács spanyolul, vagy valamilyen más közösen beszélt nyelvet használ. A filozófusokat az sem érdekli, hogy a titkárnők elektronikus levélben, esetleg faxon küldik el az életbölcsességeket. Ugyanakkor a titkárnőket sem érdeklik, hogy a tolmácsok és a filozófusok milyen nyelven beszélnek. Nekik csak az a fontos, hogy az előre megbeszélt módon (jelen esetben elektronikus levélben) célba jusson az üzenet. A kapcsolattartás tehát úgy is sikeresen végbement volna, ha a titkárnők faxot, a tolmácsok pedig valami ősi bibliai nyelvet használnak.
A hálózati program is valahogy így működik. A filozófus jelképezi a legfelső (felhasználói) szintet. Őt nem érdekli más, csak hogy az üzenete célba érjen. Ezért egy úgynevezett csatolófelület segítségével kapcsolatba lép a tolmáccsal, elmondja neki az üzenetet – ezzel az ő feladata véget is ért.
A titkárnő ebben a modellben a legalsó réteget képviseli. Az ő csatolófelültére „csatlakozik" a tolmács, aki már a semleges nyelvre lefordított üzenetet adja át.

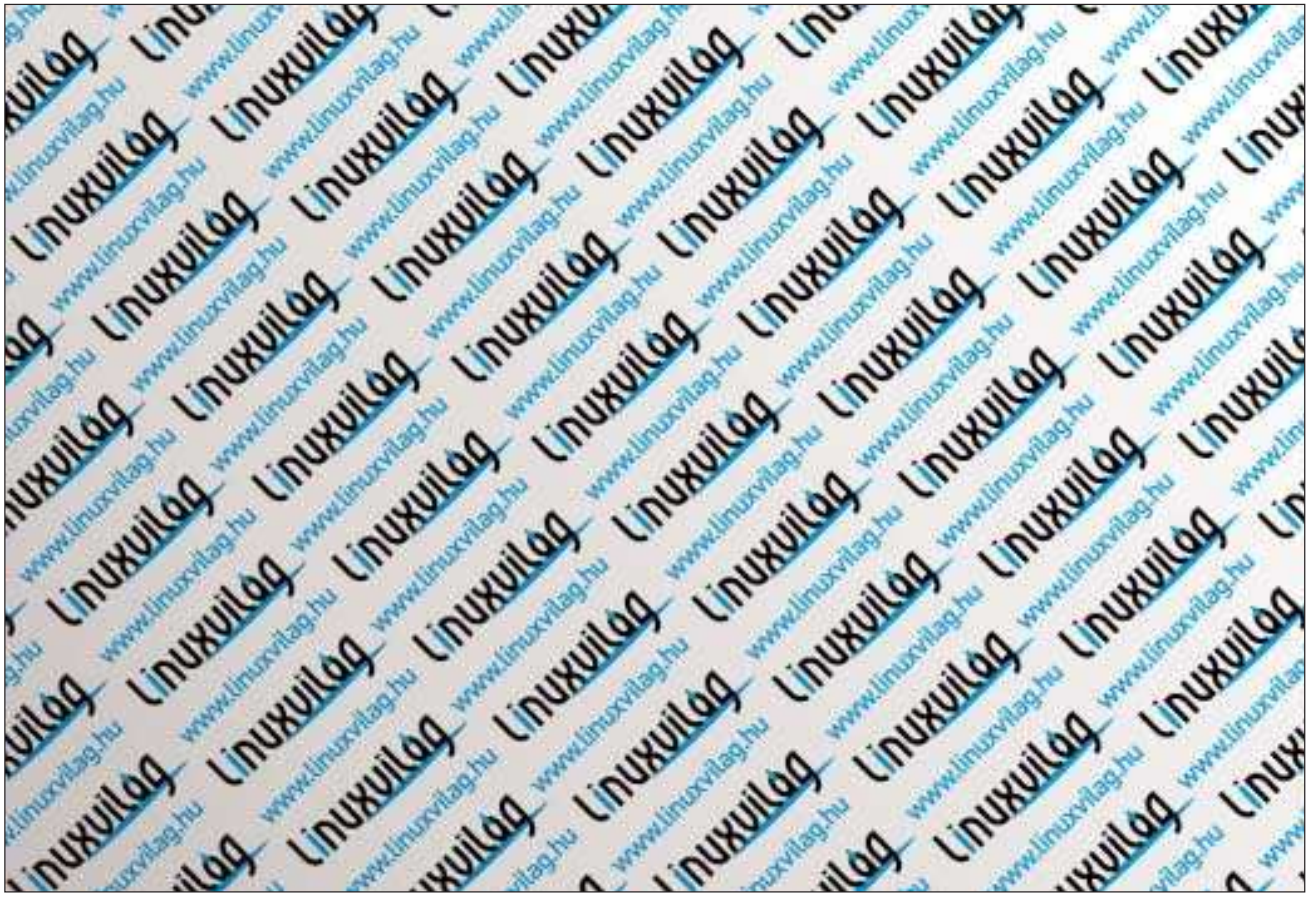
Figyeljük meg, hogy a rétegek mindig csak a velük azonos szinten lévő rétegekkel állnak kapcsolatban: a filozófus a másik filozófusnak, a tolmács a másik tolmácsnak, a titkárnő pedig a másik titkárnőnek küldi az üzenetet. (Az más kérdés, hogy a két tolmács között az üzenet átmegy még két titkárnőn, a német filozófus előtt meg két tolmácson és két titkárnőn). A titkárnők azonban sosem beszélnek a másik fél tolmácsával, és a tolmácsok sem beszélnek a másik fél titkárnőjével.
Ahhoz, hogy az azonos rétegek beszélgethessenek, szükség van olyan megállapodásokra, amelyek rögzítik a párbeszéd szabályait. Ilyen például az, hogy a tolmácsok egymás között a spanyol nyelvet használják, a titkárnők pedig elektronikus levélen keresztül továbbítják egymásnak az adatokat. Ezeket a megállapodásokat protokolloknak nevezzük.

Hálózati kiépítések

Az első ilyen dolog a csatolófelület, amelyről azt mondtuk, hogy a két réteg közötti kapcsolatot valósítja meg. A gyakorlatban a csatolófelület nem más, mint elemi műveletek halmaza. Azokat az elemi műveleteket tartalmazza, amelyeket az alacsonyabb réteg a közvetlenül felette lévő réteg számára el tud végezni.
A rétegek és a rétegek által használt protokollok pontosan meghatározzák a hálózatot, ezért őket együttesen hálózati kiépítésnek nevezzük. A hálózati kiépítésbe azonban nem tartozik bele, hogy a rétegeket miként valósítják meg (implementation). Ugyanígy nem foglalkozik a csatolófelületek meghatározásával. Ezek mind-mind a programozóra vannak bízva, aki megírja az adott réteghez tartozó programot, illetve (a fizikai rétegek esetében) elkészíti az adott eszközt.

Végül nézzünk egy példát egy üzenet elküldésére, hátha ezzel megvilágíthatjuk az esetlegesen sötéten maradt foltokat. Tegyük fel, hogy az adott hálózati kiépítés négy rétegből áll. A legfelső rétegbe mindig a felhasználói alkalmazások tartoznak, így egy felhasználói folyamat az elküldeni kívánt üzenetet átadja a harmadik rétegnek. A harmadik réteg nem tesz semmiféle kikötést az üzenet hosszára nézve, mivel a felhasználóknak elég kényelmetlen lenne, ha például egyszerre csak 100 bájtot küldhetnének. A valóságban azonban nem lehet bármekkora adatot egyszerre átvinni, ezért a második rétegnek meg kell határozni valamiféle legnagyobb méretet.
Ha az üzenet nagyobb, mint ez a legnagyobb méret, akkor a harmadik rétegnek fel kell darabolnia. A darabok elé egy fejlécet illeszt, amely elárulja, hogy ez az üzenet hányadik darabja. Ez azért hasznos, mert egyik rétegnek sem kell figyelnie arra, hogy az üzenetdarabokat sorrendben továbbítsa. Ezenkívül nagyobb hálózat esetén hétköznapi jelenség, hogy a célállomásra nem a küldés sorrendje alapján érkeznek meg az adatok.
A második réteg – ha úgy gondolja – további adatokat csatolhat a csomaghoz, majd átadja a legalsó rétegnek, amely a fizikai továbbításért felelős. Amikor az üzenet megérkezik a célállomásra, a második réteg „leszedi” az általa csatolt adatokat, majd küldi tovább a harmadiknak. A harmadik a darabokból összeállítja a teljes csomagot (előtte mindegyikről letakarítja a fejlécet), majd továbbítja a legfölsőnek. Láthatjuk, hogy az üzenethez extraadatot csak az alsó rétegek csatolhatnak, és ezek az adatok nem kerülnek a célállomás felsőbb rétegeihez.

Garzó András (garzoand@interware.hu)



Hogyan térjünk át Linuxra lépésről lépésre?

Második lecke: vegyük birtokba a Linuxunkat!

Mint azt az előző részben megígértem, folytatjuk az átállást a grafikus felület által rendelkezésünkre bocsátott ***Vezérlőközpont*** és a terjesztésfüggő YaST2 bemutatásával. Az utóbbinál természetesen nem hagyhatjuk ki az alapvető csomagkezelési ismeretek megszerzését sem. Mindenekelőtt azonban nem árt megismerkednünk azzal a grafikus felülettel, amelynek segítségével rendszerünket időnk 99 százalékában használjuk. Írásunkban folytatjuk az előző részben megkezdett SuSE-vonalat, ami azt jelenti, hogy a rendszer összetevőinek működését ezen a változaton – mint példán – keresztül ismerheti meg az olvasó. Vágjunk bele!

A grafikus felület

Mint a legtöbb korszerű operációs rendszer esetében, mára a Linuxnál is alapértelmezetté vált a grafikus felhasználói felület használata. Ez arra hivatott, hogy a felhasználó számára – kényelmes beavatkozási felületet nyújtva – megkönnyítse az amúgy körülményes géphasználatot. Szeretném azonban megjegyezni, hogy – terjesztéstől függetlenül – a grafikus felületek használata egyáltalán nem kötelező, nincs szorosan és elmozdíthatatlanul beleépítve a rendszerünkbe, a későbbiek során is bármikor cserélhető, frissíthető, kikapcsolható. Gondoljunk csak a kiszolgálókon történő futtatásra. Ilyen esetekben például semmi szükségünk nincs különböző erőforrásokat lefoglaló felületekre. Most maradjunk a hagyományos használat mellett, ugyanakkor nézzünk egy kicsit a dolgok mögé. Először is vonatkoztassunk el az eddig megszokott, egységbe zárt gondolkodásmódtól, és képzeljük el Linuxunkat úgy, mint egy hatalmas, több tízezer darabból álló legót. Valamilyen szinten az összes operációs rendszer hasonló felépítésű, annyiban különböznek egymástól, hogy az egyes elemek „gyárilag" össze vannak ragasztva, hogy ne lehessen őket szétszedni, de sokszor még ezek az összeragasztott darabok is nagyobb darabokká állnak össze, a felhasználó által nem szétbontható módon. Mivel az olvasó többnyire ilyen rendszerekkel találkozhat, nem meglepő, ha ez a legtöbb embernek már fel sem tűnik. Megsúgom, a tervezőknek pontosan ez volt a szándékuk: elrejteni a működés részleteit és a felhasználók számára átlátszóvá tenni a gép működtetését.

A Linux esetében azonban ez a bizonyos építőjáték nincs ennyire összekovácsolva, az egyes szerkezeti egységek egymáshoz hozzáfűzhetők vagy épp egymástól leválaszthatók. Ilyen szerkezeti egység a grafikus felület is, amely – mint már említettem – alapértelmezetten hozzá van fűzve telepített rendszerünkhöz, önműködően ez indul el a gép bekapcsolását követően, és mi, felhasználók már csak ezt látjuk. Ennek ellenére a Linux a legtöbb szolgáltatásával együtt enélkül is vígan fut, a felület csais a mi munkánkat segíti, használatával elérhetjük a Windows operációs rendszereknél megszokott kényelmet és használhatóságot. Most megkérdezhetnénk, hogyha úgyis a kényelmet hivatott elősegíteni, akkor miért nincs szorosan összekapcsolva a rendszerrel olyan módon, mint redmondi társai esetében? A válasz egyszerű: így szélesebb körben használhatjuk a rendszerünket. Ez a széles kör arra is vonatkozik, hogy szélesebb felhasználói réteg igényeit is ki lehet ilyen módon elégíteni. Valós példával élve: Linux alatt felhasználói felületünket bármikor kicserélhetjük egy másfajtára. Például ha nekünk csak bizonyos grafikus szolgáltatásokra van szükségünk, akkor választhatunk egy olyan felületet, amely azáltal, hogy nem az összes szolgáltatást nyújtja, kis erőforrásigényével gyakran igen jelentős sebességnövekedést el tud érni. Nem is kell azonban ilyen messzire mennünk, elég, ha valaki csak a meggyőződésének engedelmeskedve inkább egy másik felhasználói felületet szeretne használni, az általa nyújtott lehetőségeket igénybe venni. Ezeken túlmenően a kínált szolgáltatások tekintetében is előnyökkel jár az effajta rendszerfelépítés, ám ezeknek az előnyöknek az ecsetelését terjedelmi okokból kihagynám, s rátérnék a grafikus felületek ismertetésére. Látni fogjuk, hogy grafikus felületünk is több alkotóegységből áll, csakúgy, mint maga a Linux, s a későbbiekben az is nyilvánvaló lesz számunkra, hogy annak minden alkotóegysége újabb alkotóegységekből áll. Ne féljünk azonban ettől, nekünk az alkotórészeket nem kell külön-külön ismernünk – a segítségünk nélkül is remekül elboldogulnak. Mindez mégis azért jó a számunkra, mert így szinte minden apró hajszálnyi részletet lehetőségünk nyílik beállítani. Gyakorlatilag a végletekig testreszabhatjuk rendszerünket mind a külcsín, mind a működés tekintetében.

Az X-felület

Annak érdekében, hogy az előzőekben ecsetelt cserélhetőség során rendszerünk ne szenvedjen csorbát, mindenekelőtt egy összefogó program használatára van szükség, amelynek a használata egyrészt szabályossá teszi a felületek működését, másrészt alapértelmezetten nyújt olyan szolgáltatásokat, amelyeket úgyis minden felhasználói felület használni fog. Ilyen például maga a grafikus képernyő. Az X-felület pontosan ezt teszi: termőtalajt biztosít a felette futó felhasználói felületek számára. A program alapvetően egy kiszolgáló, amelyre helyi vagy távoli gépről is kapcsolódhatnak más programok, s az X-kiszolgáló dolga az, hogy az így kapcsolódott programokat grafikus felületén a felhasználó számára megjelenítse. Esetünkben a kiszolgálóhoz a saját gépünkről, helyből csatlakozik a felhasználói felületet működtető program. Amit mi, felhasználók végül látunk, s amit használunk, az ez a működtető program. A SuSE 8.2-ben (a korábbi és feltehetően későbbi változatokban is) az alapértelmezett felhasználói felület a KDE, amely a fent említett Linuxban a 3.1-es változatnál jár.

Mi is ez a KDE?

Ha a rövidítés rejtette kapcsolatot lefordítjuk, azt kapjuk, hogy K-munkakörnyezet (K Desktop Environment), amelynek már a nevéből is kitűnik, hogy a fentebb vázolt felhasználói felületeken egy kissé túlmutat. Nemcsak az alapvető kezelési szolgáltatásokat valósítja meg, s teremt számunkra egységes felületet, de nagyszámú alkalmazást fejlesztettek hozzá, számos program támogatja különleges lehetőségeit, és a Windowshoz hasonlóan beépített böngészővel, beépített ablakkezelővel bír, immáron az összevontság jegyében.

Amikor sorozatunk előző részében „grafikus felületet” emlegettem, erre a bizonyos programra utaltam, amit a bejelentkezés után láthatunk. Nem kell különösebben éles szem ahhoz, hogy észrevegyük: ez bizony kísértetiesen hasonlít a Windows felületére. (Akit ez feszélyez, kedvére átszabhatja.) Most érkeztünk el ahhoz a ponthoz, amikor meg kell említenem, hogy ezért ne is féljünk a használatától. Amit eddig leírtam, talán bonyolultnak tűnhet, de mindez csupán tájékoztató,

1. kép A SuSE súgója

2. kép A YaST vezérlőközpont

fejtágító jellegű, nem kell feltétlenül ismernünk. Ha viszont megértjük az alapokat, a későbbiek során könnyebben kihámozzuk az esetleges eltérések, működési sajátosságok miértjét, megtanulhatjuk az összetevők testreszabási lehetőségeit, s ezáltal hosszú távon igazi gurukká válhatunk.

Az emelkedett szöveg után térjünk vissza egy kicsit a nyers valósághoz. A következő dolog, amitől a kezdő felhasználó megijedhet, az, hogy ezen a felületen annyi mindent beállíthat, kiszínezhet, bekapcsolhat, áthelyezhet, hogy nem igazán tudja, mihez is fogjon. Azt javaslom, mindent csak a maga tempójában tegyünk, ne ijedezzünk, hanem kezdjük rögtön a legelején. Nem árt, ha ezeket a testreszabási lehetőségeket összefogottan kezeljük, így elkerülhetjük, hogy a sok kapcsoló között eltévedjünk. A KDE alatt akad egy eszköz kifejezetten erre a célra, a ***Vezérlőközpont***, amelynek segítségével rendszerünket átlátható módon kezelhetjük.

A Vezérlőközpont

A ***Vezérlőközpont***-ot a KDE menüben (***Start*** menü) található hivatkozással indíthatjuk. Ha az ablak megjelent, a bal oldalon egy kategóriamenüt találunk, amely típus szerint rendszerezi a beállítási lehetőségeket. Bármelyikbe belépve a legelső menüpont egy „vissza” hivatkozás lesz, ennek révén úgy érezhetjük magunkat, mintha egy könyvtárszerkezetben lennénk. Számos almenü ugyanígy almenüket tartalmazhat, amelyekre ugyanazok a szabályok érvényesek, mint a főmenüre. Ha egy menü nem almenü, hanem egy beállítási pont, akkor a jobb oldali nagy ablakrészben egy panelt kapunk, amely az adott ponton rendelkezésünkre álló beállítási lehetőségeket tartalmazza. Minden esetben, amikor egy másik beállítási ponthoz szeretnénk vándorolni, a beállításokat még azelőtt mentsük, mielőtt az új panel ablakunk jobb oldali részébe betöltődne – erre természetesen a program is figyelmeztet. Számos beállítópanel jobb alsó sarkában szerepel egy gomb: a ***Rendszergazdai mód***. Ezt megnyomva rendszergazdaként jelentkezhetünk be a program használatához, így a különleges, érzékeny beállítások megváltoztatására is lehetőségünk nyílik (például a bejelentkező képernyő megváltoztatására, a dátum átállítására).

A bal oldali menürendszer felett három fület találhatunk. Alapértelmezetten mindig az első aktív, a ***Keresés*** fülre kattintva azonban egy érdekes megoldást tanulmányozhatunk: lehetőségünk nyílik címszavak keresésére, kijelölésére. Ezek után a szavak listája alatt megjelennek azok a beállítási pontok, amelyeken belül az adott szó által képviselt témakörre vonatkozó beállítási lehetőség szerepel. A harmadik fül alatt mindig a pillanatnyilag betöltött beállítási ponthoz tartozó segítséget találjuk, természetesen magyarul. Most pedig nézzünk meg néhány, nekünk fontos beállítási lehetőséget.

Grafikai megjelenés

Ezen belül találhatunk minden olyan beállítást, amely a felületünk kinézetét befolyásolja: megváltoztathatjuk az ablakok keretezését, az alkalmazások indítása során történő visszajelzést, a megjelenő betűtípusokat, asztalunk hátterét, lecserélhetjük az ikonkészleteket, képernyővédőket állíthatunk be, módosíthatjuk a KDE-menü elemeit, a felhasználói paneleken található vezérlőelemek (például a lenyíló menü) stílusát és a grafikus rendszer színösszeállítását. Ne feledkezzünk meg róla, hogy egy csomó beállítási pont panelje további füleket tartalmaz, és ezeken belül újabb beállítási lehetőségeket találunk.

KDE-összetevők

Itt találhatjuk meg a felhasználói felületbe épített különböző programok beállításait. Ez a lista attól függően is változhat, hogy milyen rendszerösszetevőink vannak telepítve. A listából kiemelném ***Fájlkezelő*** beállításait, amellyel a Windowsban megszokott ***Sajátgép***-re kattintva az előugró fájlböngésző linuxos megfelelőjét (Konqueror) szabhatjuk testre. A Konqueror egyébként egyben a rendszer webböngészője, ami szintén nem újdonság (viszont az internetböngészésre vonatkozó beállítások máshol találhatók).

A ***Fájltársítások*** fül egyelőre még nem fontos, de a későbbiekben itt állíthatjuk majd be, hogy milyen típusú fájlt milyen paranccsal nyisson meg a rendszerünk. Néha csupán tájékozódás végett is használhatjuk, ha egy fájlról meg szeretnénk tudni, hogy a mi rendszerünkben támogatott-e. A típusát ebből a listából kikeresve könnyen meggyőződhetünk róla.

A ***Munkafolyamatok*** beállítási ponton belül a bejelentkezés utáni szabályokat állíthatjuk át. Megmondhatjuk, hogy bejelentkezés után milyen állapotba szeretnénk visszatérni (akar-

juk-e használni a legutóbbi kikapcsolás során mentett állapotot stb.), valamint kijelölhetjük az alapértelmezett műveletet arra nézve, hogy mi történjék közvetlenül a kijelentkezés után. Eldönthetjük például, hogy kikapcsoljuk-e a gépet, vagy inkább újraindítjuk, esetleg csak másik felhasználónak szeretnénk átadni, s az ehhez szükséges bejelentkező képernyőre van szüségünk.

Munkaasztal

Beállíthatjuk az ablakműveleteket: hogyan nagyítsuk, mozgassuk, zárjuk be, miként fókuszálhatunk rá és még sok egyéb mást. Ezeken kívül az alsó panelen módosíthatjuk az elindított programok listájának megjelenését, befolyásolhatjuk az asztal működését, megjelenésének a tulajdonságait, részletesen beállíthatjuk az alsó panel – amelynek a neve ***Panel*** – tulajdonságait, megváltoztathatjuk az asztal méretét. Kezelhetjük továbbá az alapértelmezetten rendelkezésünkre álló ***Virtuális munkaasztal*** szolgáltatásait, amely lehetővé teszi számunkra, hogy egy monitoron egymás alatt több ugyanolyan munkaasztalunk is legyen. Az ilyen munkaasztalok között váltva az az érzésünk támad, mintha minden esetben másik monitor elé ülnénk. Ez akkor hasznos, ha rengeteg ablakot használunk egyszerre.

Külső egységek

Ezek közé sorolhatjuk például a billentyűzetet, a digitális kamerát, az egeret és a nyomtatót. Külön említést érdemel a ***Nyomtató*** panel, amely az eddig megszokottakkal ellentétben összegzi a nyomtatás során elérhető szolgáltatásokat, a különböző nyomtatómeghatározásokat stb. Fontos megjegyezni, hogy a faxolás vagy a fájlba nyomtatás műveletet is itt tudjuk befolyásolni, hiszen a rendszer működése szempontjából ez hasonlóképpen zajlik, mintha valóban nyomtatóra dolgoznánk.

Rendszerfelügyelet

Az összes beállítására jellemző, hogy csak rendszergazdai módban módosíthatjuk. Például itt állíthatjuk át a bejelentkezés során felpattanó képernyőt, a bejelentkezés folyamatának a tulajdonságait, kiiktathatjuk magát a bejelentkezést, testreszabhatjuk a panel kinézetét. Ezeken kívül itt változtatható meg a rendszeridő, új betűtípusokat adhatunk rendszerünkhöz, módosíthatjuk az elérési utakat, bár a kezdő felhasználóknak ezt egyáltalán nem javasolnám.

YaST

Ami a KDE számára a vezérlőközpont, az a rendszerünk számára nem más, mint a YaST. Már a ***Vezérlőközpont*** egyik menüpontjaként is elérhető a YaST modulok listája, ám ez idáig nem tudtuk, hogy mindez mit is takar. A YaST a SuSE által fejlesztett rendszervezérlő és karbantartó program. Ez fogja össze magát a terjesztést. Feltétlenül meg kell néznünk részletesen, hogy mit is tesz.
Az előző cikkben már szóltunk róla, ugyanis magát a telepítést is a YaST irányítja, de ezeken túl kezeli a hálózati eszközöket, a számítógép eszközeit, a rendszer szolgáltatásait és a programösszetevőket.
Nézzük most egy kicsit részletesebben az egyes modulokat!

Biztonság és felhasználók

Mint már említettem, ez többfelhasználós rendszer, ami azt jelenti, hogy egyidőben egyszerre többen is használhatják gépünket. A folyamat szabályozása érdekében minden embert, aki használni szeretné, meg kell ismertetnünk a rendszerünkkel. Mindez szaknyelven annyit tesz, hogy felhasználót kell létrehoznunk

3. kép A KDE Vezérlőközpont

a gépen, amely a használat során az adott illetőt képviseli. Ebben a modulban hozhatunk létre új felhasználókat, új felhasználói csoportokat, és itt módosíthatjuk a tűzfalunk beállításait, ezekre azonban egy későbbi részben térnék ki külön. A tűzfal egyébként arra szolgál, hogy megakadályozza a hálózat (internet) irányából a számítógépünkre történő illetéktelen behatolást.

A gép

A ***Hardver*** alatt állíthatjuk be gépünk összetevőinek sajátosságait a CD-ROM-meghajtótól kezdve a hangkártyán keresztül a lapolvasókig mindent beleértve. Használata és feladata nagymértékben hasonlít a Windowsban megszokottakhoz, valamint érvényesek rá az előző cikkünkben kimondott szabályok.
Az összetevőt kiválasztva minden esetben egy listához jutunk, ahol a felismert eszközöket látjuk, vagy egyéb esetben (például az egér, amely minden géphez tartozik) egy listát kapunk a használható meghajtóprogramokról. A lista megfelelő elemére kattintva szabhatjuk testre az adott eszközt.

Hálózati eszközök

Számtalan hálózati kapcsolatot biztosító eszközfajta közül választhatunk ebben a modulban, s lehetőségünk nyílik kényelmesen beállítani kapcsolatunk jellemzőit. A SuSE által támogatott hálózati kapcsolatfajták: ***DSL***, ***fax***, ***hálózati kártya***, ***ISDN***, ***modemes kapcsolat***.

Hálózati szolgáltatások

A Linux számtalan hálózati szolgáltatás biztosítására képes a webkiszolgálótól a levélkiszolgálókon keresztül a fájlkiszolgálókig bezárólag. Minden egyes szolgáltatás egyedi jellemzőkkel bír. Ezeket a jellemzőket állíthatjuk be ebben a modulban. Kezdők számára ez egyelőre homályos terület, hagyományos használat során ritkán kell idekeverednünk.

Programok

A ***Szoftver*** igen fontos modul a későbbi használat szempontjából, ugyanis itt gondoskodhatunk rendszerünk programállományának a karbantartásáról. Frissíthetjük a rendszert a SuSE hálózati kiszolgálóiról vagy frissítő CD-iről. Ekkor a jelenlegi programok javított, jobb, biztonságosabb változatai kerülnek önműködően a programok közé. Ha nincs széles sávú kapcsolatunk, akkor elégedjünk meg a jelenlegi rendszerünkkel, és ne akarjuk frissíteni. A legfontosabb lehetőség azonban a ***Szoftver telepítése és eltávolítása***, ám ennek megértéséhez tegyünk egy kis kitérőt.

Csomagkezelés

Minden terjesztés egymáshoz kapcsolódó csomagok ezreiből épül fel. Ezeknek a csomagoknak a karbantartását, telepítését, eltávolítását, frissítését, egységbe szervezését nevezzük csomagkezelésnek. Talán könnyebb lesz megérteni, ha gondolatban telepítünk egy csomagot. Linux alatt az egyes programok az esetek jelentős részében csomagok formájában állnak rendelkezésre. Egy ilyen csomag windowsos megfelelője egy telepíthető *.exe* állomány, amely indítás után az általunk meghatározott helyre másolja magát, beállítja a saját jellemzőit, azután fut. Eltávolításakor a telepítőprogram inverze hívódik meg, amely törli a programot, de általában sok szemetet hagy maga után. Linux alatt a csomag tehát egy ilyen exefájl megfelelője, azzal az igen fontos különbséggel, hogy a telepítőprogramokkal ellentétben a csomagok nem élhetnek önálló életet, létüket a terjesztés központi csomagkezelő mechanizmusa szervezi rendszerbe.

4. kép Vezérlőközpont–címszavak szerinti keresési lehetőség

Egy csomag tartalmazza, hogy hová kell települnie, milyen alapbeállításokkal kell elindulnia, milyen más csomagokra van szüksége a működéséhez (ezt nevezzük függőségnek), hogyan távolítható el stb. Ha egy csomagot telepíteni szeretnénk, nem kell tudnunk róla semmit, nem kell ismernünk, egyáltalán nem kell, hogy értsünk a tartalmazott program felépítéséhez. Ahhoz, hogy működésre bírjuk, meg kell kérnünk a csomagkezelőt, hogy ágyazza be a csomagot a rendszerbe és tegye lehetővé a használatát. Ha netán hiányozna egy olyan csomag, amely nélkül a telepíteni kívánt egység nem működhet, a csomagkezelő figyelmeztet bennünket, és addig nem telepíti a programot, amíg rendelkezésére nem bocsátjuk a szükséges másik csomagot. Ilyen esetekben azonban a csomag letöltési helyén fel van tüntetve, hogy milyen függőségei vannak, s legtöbb esetben onnan le is tölthetők. Ezzel a csomagkezelési módszerrel egy karbantartható központosított rendszert kapunk, amely szigorú összevontságával biztosítja rendszerünk épségét, és ennek is köszönhetően nem „szemetesedik el" egykönnyen. Ezentúl a helyesen elkészített csomagok jelentősen megkönnyíthetik a felhasználók munkáját.

Vissza a YaST-hoz

A SuSE az RPM nevű csomagkezelő rendszert használja. Ez a program tárolja a telepített csomagok listáját, sok-sok jellemzőjüket, s ezek alapján, valamint a csomagban tárolt adatok figyelembevételével telepíti az egyes csomagokat. A YaST programtelepítő modulja ennek a csomagkezelőnek egy előlapja (frontend), amely a csomagok profi módon történő kezelését teszi lehetővé: a CD-kről rendelkezésre álló összes csomag kategorizált megjelenítését, kereshetőségét, adatokat szolgáltat a csomagokról, telepíti, illetve eltávolítja az általunk kijelölt összetevőket, így nekünk a csomagkezelés részleteiről szinte semmi sem kell tudnunk.

A programkezelő modul felépítése

Az ablak bal oldali menüjében egy csoportlistát láthatunk, erre rákattintva a jobb oldalon megjelennek az ebbe a csoportba tartozó csomagok. Amelyik előtt üres jelölőnégyzet áll, az pillanatnyilag nincsen telepítve; ami pipával jelölt, az már a rendszer részét képezi. (Egyéb jelölésekért lásd a ***Szűrő*** lenyíló menü ***telepítési összegzés*** pontját.)
A jobb oldali ablakrész alján található az adott csomag adatpanele, amely a csomag teljes nevét, rövid leírását tartalmazza, a többi fülön belül pedig a technikai adatokat, a függőségeket és a változatokat tünteti fel. Innen tájékozódhatunk az adott program feladatait illetően.
Ha a ***Szűrő*** lenyíló menüjének következő elemére kattintunk, egy másik csoportosítás szerinti csomaglistát kapunk, amelyben az ablak jobb oldali részében az adott kategóriákba tartozó csomagok jelennek meg.
Szenteljünk most egy kis időt a keresésre, ugyanis ez az egyik legfontosabb és legtöbbet használt szolgáltatás. A bal oldali keresőpanelben egy cikkszó megadása után kereshetünk a csomag nevében, leírásában, de aszerint is, hogy az adott cikkszóhoz tartozó csomag milyen más csomagokat igényel, vagy épp mit tartalmaz. A keresés eredménye pár másodperc alatt látható: az összes olyan csomagot kilistázza, amelyik illeszkedik a feltételekre. Az eredményt ugyanúgy értelmezzük, mint az eddigiek során.

Egy csomag telepítése

Példaképpen telepítsünk egy csomagot. Legyen ez a ***Csomagcsoportok*** szűrő alatt a ***Játékok*** kategória ***Logikai*** alkategóriájának ***xtetris*** nevű csomagja. Ha nem találjuk, kíséreljük meg a kereső segítségével előszedni. Tegyünk egy pipát a jelölőnégyzetbe, majd kattintsunk a ***Tovább*** gombra. Ha ilyenkor a kiválasztott csomagnak függőségei vannak, a YaST figyelmeztet bennünket, és felajánlja telepítésre a szükséges csomagokat – egyelőre minden esetben fogaduk el őket.
Ezek után a Linux követelni fogja tőlünk a negyedik korongot. Teljesítsük az óhaját, majd ha a telepítés készen van, az ALT+F2 billentyűkombinációval előhívott ***Futtatás*** panelen adjuk ki az `xtetris` parancsot, és élvezzük munkánk gyümölcsét.

Hogyan tovább?

Ha már idáig eljutottunk, elmondhatjuk, hogy értünk valamicskét a Linuxhoz. Önigazolásképpen nem árt, ha néhány ilyen ártalmatlan csomaggal (mint például a játékok) kísérletezgetve megtapasztaljuk a csomagok telepítésének minden apró fortélyát. A későbbiek során úgyis meggyűlik majd vele a bajunk, ha nem akaródzik rátalálni arra a csomagra, amire szükségünk van. Ezzel a gyakorlással egyébként néhány – a leírás alapján számunkra kedves, hasznos – csomaggal bővíthetjük rendszerünk tudástárát.
A következő cikkben bemutatásra kerülő felhasználási területek téma során számos csomagjavaslatot találhatunk majd a cikkekben, amelyeknek bízvást használ vehetjük.

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

Linuxos kiszolgálót mindenkinek!

A SuSE Linux mint kiszolgáló – kisvállalati és otthoni környezetben.

Linuxos kiszolgáló, Linux-rendszerek – fogalmak, amelyek hallatán a rendszerüzemeltetéssel foglalkozók egy része dicshimnuszokat zeng a rendszer tudásáról, sokoldalúságáról; míg másoknak rögtön az ugrik be, hogy nem működik rendesen, nincs hozzá megfelelő támogatás, állandóan foltozni kell, és akkor is csak a baj van vele. Nos, én az utóbbi álláspont képviselőit szeretném ennek az ellenkezőjéről meggyőzni. Be szeretném mutatni, hogy egy rendszer, ami akár ingyenesen is elérhető, miként tudja átvenni a kis- és középvállalatoknál használatos rendszer szerepét, és igyekszem megmutatni, hogy ez idő szerint ez miért nem is olyan bonyolult feladat.

A rendszer

Bemutatandó rendszerként a SuSE 8.2 Professionalt választottam, mert teljes magyar fordítás létezik hozzá, viszonylag friss csomagokat tartalmaz, felhasználóbarát felületen kezelhető, és számtalan eszközt ismer.
A rendszer kereskedelmi változatban is hozzáférhető – ekkor két DVD-t és öt CD-t, valamint egy vaskos dokumentációt kapunk –, illetve letölthető az internetről a ➲ http://www.suselinux.hu oldalon keresztül. Az utóbbi esetben egy telepítőkorongot kell letöltenünk, erről indítanunk a rendszert, és maga a telepítés már az internetről zajlik.

Az első lépések

A rendszer telepítése a manapság megszokott módon zajlik: helyezzük a CD-t vagy a DVD-t a meghajtóba, és indítsuk el a gépet. Ezután válasszuk a telepítést, és már töltődik is a Linux-rendszermag.
Tipp: a SuSE-CD rendszerindításakor a megjelenő képernyőn az F2 gombbal állíthatjuk át a szöveges üzemmód felbontását. Célszerű már a telepítéskor gondolni – a kiszolgálóhoz kapcsolódó monitorunk függvényében – a felbontás beállítására, ugyanis ha például egy 17"-os monitoron telepítjük a rendszert, és utána egy 14"-os régi, mondjuk VGA monitort rakunk a kiszolgálóhoz, kellemetlen meglepetésként érhet bennünket, hogy a monitoron semmi nem fog látszani, mert nem tudja kezelni a nagy felbontást.

A telepítő az indulása után rögtön felteszi nekünk a kérdést, hogy a továbbiakban milyen nyelven szeretnénk az operációs rendszert használni. Mivel a magyar nyelvet választottam, a továbbiakban a magyar telepítő kifejezéseit használom. A nyelv kiválasztása után új rendszer telepítésére, a meglévő frissítésére, sőt a sérült rendszer javítására is lehetőségünk nyílik.
Új telepítés indítása után a ***Telepítési beállítások*** párbeszédablak jelenik meg, amelyben a rendszer alapvető beállításait végezhetjük el. Mi most csak a kiszolgálók telepítéséhez szükséges beállításokat soroljuk fel, a bővebb ismertetés a Linuxvilág magazin 2003. novemberi számában található.

Az fájlrendszer

Ahhoz, hogy a kiszolgálónk megbízhatóan működjön, és az adatainkat biztonságban tudjuk, a Linux operációs rendszerek gép és program által megvalósított RAID-megoldásokat egyaránt támogatnak. A gép által megvalósított RAID-megoldások előnye, hogy azt egy céleszköz kezeli, s a programok (így az operációs rendszer) számára ez már átlátszó megoldás, mindössze a céleszköz megfelelő vezérlőprogramjainak betöltéséről kell gondoskodnunk. A program által megvalósított RAID kezelése az operációs rendszer feladata, így a különböző ope-

Letöltési kimittud

Ha az internetes telepítést választod, figyelj oda arra, hogy élő internetes kapcsolatra lesz szükséged a telepítés során.
A telepítés előtt tájékozódj a hálózatos telepítés pontos igényeiről, nehogy telepítés közben érjen meglepetés!
A telepítés folyamán egy forráshely is kívánatos, ahonnan a telepítő majd letölti a csomagokat. Ehhez az ➲ ftp://ftp.suselinux.hu FTP-elérést javaslom, mert magyar a kiszolgáló, ezért elég jó az elérhetősége. A 8.2-es változat a kiszolgálón a */pub/suse/i386/8.2* könyvtárban található.
Tipp: internetes telepítéskor mindenképpen a legkisebb méretű telepítést használd, és utólag add hozzá a további csomagokat a rendszerhez, mert ezzel jócskán meggyorsíthatod a telepítés menetét!

RAID-ábécé

A RAID – Redundant Array of Independent Disks – a különálló lemezek többszörös tömbbe fűzését jelenti. A módszer lényege, hogy több különálló lemezt logikai módon összefűzünk, hogy hibatűrő, illetve teljesítménynövelő tulajdonsággal bírjanak. A három leggyakrabban használt elrendezés a RAID0, RAID1, RAID5.

- RAID0: a kiválasztott lemezeket egymás után fűzi, így nagy tárterület alakítható ki. Hibatűrőképessége nincs, egy lemez kiesése esetén a rendszer elérhetetlenné válik. Adatelérés tekintetében gyors rendszer alakítható ki, hiszen az egyes lemezeken lévő adatok párhuzamosan érhetőek el.
- RAID1: tükrözésnek is nevezik. Itt két vagy több azonos méretű lemezen tároljuk az adatokat, olyan módon, hogy az adatokat minden lemez tartalmazza. Így például két lemez esetén az egyik lemez kiesését még túléli a rendszer, három lemez esetén két lemez kiesése lehetséges adatvesztés nélkül és így tovább. Az összeállítás nagy előnye a nagy adatbiztonság, hátránya a lemezek számához képest kis kapacitás.
- RAID5: ez a módszer ötvözi a RAID0 és RAID1 képességeit (ennek ellenére nem összekeverendő a RAID10-zel, ami RAID0-s lemezek tükrözésén alapul), egyszerre nyújt többszörözött adattárolást és lemezek összefűzését. A RAID5 rendszer megvalósításához legalább három lemez szükséges. Az alapelv egyszerű és éppen ezért zseniális. A logikai kizáró vagy (XOR) műveletet használja, amelynek az a lényege, hogyha két bit összehasonlításakor a két bit azonos, akkor az eredmény 0, ha különböző volt, akkor pedig 1. Íráskor az adatokat folyamatosan két lemezre írjuk, a harmadik lemezre pedig a két írt lemez celláinak XOR művelettel számított eredménye kerül. Világosan látszik, hogy a művelet sajátosságai miatt bármelyik lemez kiesése esetén a másik két lemezből meghatározható a harmadik lemez tartalma. Ebben az esetben a műveleti sebesség a párhuzamosítás, az elérhető tárterület az összefűzés, a biztonság pedig a hibatűrő kialakítás végett nagyobb.

rációs rendszerek másként kezelhetik a meghajtókat. (Vannak olyan rendszerek, amelyek az ilyen megoldásokat egyáltalán nem is támogatják.)
Olyan kisvállalati vagy otthoni környezetben, ahol az eszközök igen magas ára miatt nincs lehetőség a gép által megvalósított RAID-módszerek igénybevételére, megoldást kínálhat a Linux programból megvalósított RAID-megoldása. A SuSE 8.2 támogatja a RAID0, RAID1 és RAID5 módszerek akár programból való megvalósítását is.
Ehhez a ***Telepítési beállítások*** párbeszédablakban válasszuk a ***Partícionálás*** menüpontot. Itt lehetőségünk nyílik a SuSE telepítője által javasolt változat elfogadására, vagy saját lemezrész-összeállítás (partition table) létrehozására. Mivel mi most adataink fokozott biztonságára törekszünk, valamelyik RAID-megoldást szeretnénk alkalmazni, így válasszuk a ***Szakértői kézi partícionálás*** menüpontot. A megjelenő ablakban látszanak a merevlemezeink és az esetleg már létrehozott lemezrészek. Legalább két fizikai lemezre lesz szükségünk, hogy a RAID1-es tükrözést megvalósítsuk, illetve a két lemezen két-két lemezrész kell: egy az adattárolás számára, a másik pedig a memóriacsereterület (swap) számára. A csereterület az ajánlások szerint legalább annyi legyen, mint a fizikai memória mérete, de a rendszer alapos tervezése mellett ettől el lehet térni. A hatékony, gyors és megbízható működés érdekében a használni kívánt merevlemezeket érdemes különböző vezérlőkön elhelyezni, hiszen így az elérési sebesség is nagyobb, másfelől védve vagyunk a vezérlő meghibásodásából adódó kieséstől is.
Válasszunk ki tehát két merevlemezt, és készítsünk rajtuk két-két, páronként azonos méretű lemezrészt. Az adatoknak szánt lemezrész létrehozásához válasszuk a ***létrehozás*** gombot, majd válasszuk a típust elsődleges lemezrésznek. A ***Logikai partíció létrehozása*** párbeszédablakban válasszuk a ***Formázás nélkül*** lehetőséget és a lemezrész típusát állítsuk ***Linux Raid*** típusra, hiszen a későbbiekben ezt a területet szeretnénk a RAID tömb létrehozására használni. A terület méretét is megadhatjuk a párbeszédablakban, így itt sem kell elfogadnunk az alapértelmezett beállításokat. Ezt a műveletet hajtsuk végre a másik lemezre is, majd hasonló módon mindkét lemezen készítsük el a csereterületet, annyi különbséggel, hogy ennél a típusnál válasszuk a ***Formázás*** lehetőséget és a típust állítsuk ***swap***-ra.
Ha ezzel elkészültünk, akkor létrehoztuk a használni kívánt lemezterületet, és készen állunk a RAID-beállítások létrehozására. Ehhez a ***Szakértői partícionálás*** párbeszédablakban található ***RAID*** gombot használjuk. Először kiválaszthatjuk a használni kívánt RAID-összeállítást, majd a feljövő párbeszédablakban megtalálhatók lesznek a korábban RAID típussal létrehozott területeink, amelyeket most összerendelhetünk, olyan módon, hogy kijelöljük őket és a ***hozzáadás*** gombra kattintunk. Egyszerre csak egy tömböt tudunk kialakítani, így ha többet szeretnénk, a folyamatot meg kell ismételnünk.
Ha a tömb összeállításával végeztünk, akkor harmadik lépésben meg kell adnunk a területen kialakítani kívánt állományrendszer típusát és a befűzési pontját. Fájlrendszernek használjunk naplózó fájlrendszert, mert a rendszer esetleges leállása esetén az elkészített naplóállományból visszaállítható lesz az eredeti állományszerkezet és -tartalom. Ilyen naplózórendszer az ext3, a ReiserFS, az XFS és a JFS. Ezekről bővebb tájékoztatás található az interneten vagy akár a Linux-világ korábbi számaiban is. Amennyiben nem vagyunk tisztában az egyes rendszerek közötti különbséggel, a SuSE által ajánlott ext3 rendszer használata ugyan lassabb lehet, mint más fájlrendszereké, de a felépítése nagymértékben megegyezik a korábbi ext2-es rendszerrel, így a rendszer sérülése esetén adatainkhoz egyszerű módszerekkel hozzá tudunk férni. Azt a lemezrészt, amire a rendszert telepítjük, minden esetben a gyökérbe kell befűzni, így a befűzés helyéhez írjuk a / (perjelet). Amennyiben adataink számára hozunk létre lemezrészt, a gyökér egy alkönyvtárába tudjuk őket befűzni.
Tipp: az áttekinthetőség érdekében a lemezrészeket egy könyvtár alá, például a */mnt* vagy a */media* alá javasolt fűzni, és onnan közvetett hivatkozás (SYMBOLIC) létrehozni, ami a kívánt helyre mutat.
A beállítások elfogadásával végeztünk is a RAID tömb létrehozásával, visszatérhetünk a ***Telepítési beállítások*** párbeszédablakhoz.

A programok

Ha végeztünk a tárhely kialakításával, akkor itt az ideje, hogy tartalommal is megtöltsük: válasszuk ki a telepíteni kívánt programokat. Ezt a ***Telepítési beállítások***/***Szoftver*** menüpontjában tehetjük meg. Itt a telepítő rögtön felajánl három telepítési módot, amelyek közül mi vagy a ***Minimális rendszer***-t vagy a ***Minimális grafikus rendszer***-t válasszuk. Mivel a gépünk kiszolgálóként fog üzemelni, várhatóan mindennapi munkára (munkaállomásként) senki nem fogja használni, tehát nagy erőforrásigénye miatt a grafikus rendszert mellőzhetjük.

Szöveges felületen a SuSE minden olyan szolgáltatást elérhetővé tett, ami grafikus felület alatt ugyancsak megtalálható, így a grafikus rendszert csak akkor telepítsük, ha teljesen idegen számunkra a szöveges felület, és van felesleges erőforrásunk a kényelmi megoldásokra.
Ha kiválasztottuk valamelyik összeállítást, akkor válasszuk a ***Részletes választás*** gombot, hogy egyéb hasznos programokat, segédeszközöket telepítsünk. A gombra kattintva elindul a SuSE YaST csomagkezelő alkalmazása, ebben – több szempont szerint csoportosítva – kiválaszthatjuk a telepíteni kívánt csomagokat. A csoportosítás módja a ***Szűrő*** menünél adható meg, itt választhatunk a csomagok több szempont szerint való rendezése és a keresés között. Ha az összeválogatással végeztünk, akkor az elfogadás után a rendszer leellenőrzi a csomagok függőségeit és esetleges ütközéseit, ha talál ilyet, fel is hívja rá a figyelmünket, így módunk lesz beavatkozni a telepítés menetébe.
Tipp: kezdetnek válasszunk ki egy pár olyan csomagot, amelyeknek később, a felügyeleti munka során nagy hasznát fogjuk venni. Ilyen csomag a Midnight Commander szöveges felületű állománykezelő; a Webmin csomag, ami a rendszer jellemzőinek beállítását teszi lehetővé egy böngészőn keresztül; a `wget`, amivel a HTTP-protokollon keresztül internetes tartalmakat tudunk parancssorból letölteni; illetve a `links`, ami egy egyszerű, szöveges felületű böngészőprogram; és esetleg az `ncftp`, amely egy parancssoros FTP-ügyfél. Egyelőre több csomagot nem is telepítünk, amennyiben mégis szükségünk lenne másra, telepítésükről a használatbavétel előtt gondoskodunk.

Rendszerindítási beállítások

Következő fontos feladatunk a rendszer indítási jellemzőinek a beállítása. Ez azért szükséges, hogy a gépünk a bekapcsolás után el tudja indítani a telepített operációs rendszert. Ehhez telepíteni kell az úgynevezett rendszerbetöltőt – Linux alatt a két legelterjedtebb rendszerbetöltő a LILO és a Grub. Feladatuk, működésük közel azonos.
A rendszerbetöltő számunkra legfontosabb jellemzője a telepítési helye lesz. Ez azért kiemelt fontosságú, mert ha programból megvalósított RAID-ot használunk, a gép indításakor, vagyis a rendszermag betöltése előtt még nem áll rendelkezésünkre. Éppen ezért a rendszerbetöltőnek meg kell mondani, hogy melyik lemezen keresse majd azt az alkalmazást, aminek a segítségével indítani tudja a rendszert.
Ezeket a beállításokat a ***Telepítési beállítások*** párbeszédablakban a ***Rendszerindítás*** menüpont alatt találjuk meg. Ezt elindítva betöltődik a ***Rendszerindító*** beállítási párbeszédablaka, ahol ***a rendszerbetöltő helye*** jellemzőt kell majd módosítanunk. Nyissuk meg e kapcsoló beállításait, és a kijelölést állítsuk az ***egyéb*** mezőre, az értékét pedig a ***/dev/hda***-ra, amennyiben a lemez, amiről indítani szeretnék, az elsődleges vezérlő első helyén áll. Ennek az értéknek a helyes beállítása nagyon fontos, mert elmaradása vagy rossz beállítása esetén a gép indulásakor a rendszermag nem fog betöltődni, azaz a rendszer nem indul el.

Az utolsó lépések a telepítés indításáig

Ha az eddigiekkel végeztünk, maradt még néhány apró beállítás, és utána indulhat a másolás. Az egyik ilyen beállítás az időzóna helyes megadása – ez azért olyan fontos, mert ha például levélkiszolgálót kívánunk üzemeltetni, nagyon zavaró, amikor a rossz beállítás miatt a levelek dátumozása megváltozik, így azok korábbinak vagy későbbinek fognak tűnni, mint amilyenek valójában. Továbbá gondot jelenthet a naplóállományok rossz időbélyegezése (timestamp) is, mert így követhetetlen lesz a gép naplózása.

Ha minden apróság pontos beállításával végeztünk, elindíthatjuk a telepítést. A telepítő még egyszer rákérdez, hogy helyes-e a beállítás és tényleg telepítheti-e a rendszert, majd ezt is elfogadva indul a másolás. Az első CD felmásolása után a rendszer újraindul, és rögtön kiderül, hogy helyesen állítottuk-e be a rendszerbetöltőt, mert ha nem, már itt el fogunk akadni. Indításkor alapesetben nem kell a géphez nyúlni, az magától indítja a telepítés folytatását. Ha a szöveges felület telepítését választottuk, akkor innentől kezdve már szöveges felületen folyik a telepítés. Helyezzük be a kért lemezeket és kövessük a telepítő utasításait.

A másolás utáni lépések

A másolás befejezésével a telepítő a rendszergazda (root) jelszavát kéri. A rendszergazda jelszavának megadásával egy időben a ***Szakértő gombra*** kattintva kiválaszthatjuk a jelszavak titkosítására használt algoritmust is. Használhatunk MD5, DES és Blowfish titkosítást, illetve ellenőrző algoritmust. Amennyiben egyedül álló kiszolgálót telepítünk, úgy gyakorlatilag teljesen mindegy, hogy melyik titkosítást használjuk, ez nem lesz kihatással rendszerünk működésére. A DES titkosítás talán a legelterjedtebb titkos kulcsú titkosítási módszer, alapesetben ennek a használata javasolt.
Ha végeztünk a jelszavak beállításával, a következő lépésben hitelesítési eljárást választhatunk, amennyiben a többi gép felhasználói adatbázisa a hálózaton keresztül elérhető. Alapesetben válasszuk az ***Egyedüli gép a hálózatban*** lehetőséget.

A hálózat beállításai

Elérkeztünk ahhoz a pillanathoz, amikor el kell végeznünk a hálózat beállítását. Lehetőségünk nyílik hálózati kártyák, modemek, ADSL- és ISDN-eszközök beállítására. Mi most a hálózati kártyák és ADSL-eszközök beállításait helyezzük figyelmünk középpontjába, a modemek beállításairól a Linuxvilág 2003. novemberi számában olvashatunk, míg az ISDN-eszközökkel még a későbbiek folyamán találkozunk.
A hálózati kártyák menüpontját kiválasztva lehetőségünk nyílik a kártyák beállítására. A megjelenő párbeszédablakban láthatjuk a még nem és a már beállított eszközöket. Az eszköz kiválasztása után elkezdhetjük a beállításukat. A megjelenő párbeszédablakban először is kiválaszthatjuk, hogy a kártya beállításait önműködően egy, a hálózatunkban már működő DHCP-kiszolgálóra bízzuk-e, vagy mi magunk határozzuk meg az IP-címet és a hálózati maszkot. Miután ezzel végeztünk, a ***Gépnév és névkiszolgáló*** gombra kattintva beállíthatjuk gépünk nevét, a tartomány-, a névkiszolgálókat és a tartománykeresési listákat.

Az előző párbeszédablakhoz visszatérve az ***útválasztás*** lehetőség alatt beállíthatjuk a hálózatban található átjárókat, köztük az alapértelmezett átjárót. Továbbá ebben a párbeszédablakban engedélyezhetjük a hálózati kártyák között az IP-továbbítást, így kiszolgálónk a hálózat többi gépe átjárójának a szerepét játszhatja. Ezeket a beállításokat a jelenlegi vagy a kialakítani kívánt hálózati szerkezet alapján végezzük el.
Amennyiben ADSL-kapcsolattal rendelkezünk, úgy a ***DSL kapcsolatok*** menüpontjára kattintva ezt is rögtön beállíthatjuk. Ebben a párbeszédablakban, amennyiben a telepítő nem fedezett fel ADSL-kapcsolatot, kézzel is létrehozhatjuk azt. Új kapcsolat hozzáadásakor ki kell választanunk a PPP-módot, ez jelenleg hazánkban általában a PPP over Ethernet, azaz PPPoE. Ezután azt a hálózati kártyát is meg kell adnunk, amihez az ADSL-modem csatlakozik. Ehhez a hálózati kártyához szükségtelen IP-címet hozzárendelni, hiszen ezt a kártyát hálózati kapcsolatra nem fogjuk használni, ugyanis ebben az esetben a PPP-kapcsolathoz lesz az IP-cím és a hálózati maszk rendelve. Miután megadtuk a hálózati eszközt, kiválaszthatjuk a kapcsolat kiépítésének a módját, ami lehet önműködő az eszköz bedugásakor (Hot Plug), önműködő rendszerindításkor, illetve kézi indítású. A kapcsolatot kézi indításnál a `cinternet` parancs megfelelő paraméterezésével indíthatjuk a parancssorból.
Következő lépésben internetszolgáltatónk adatait kell megadnunk. Válasszuk az ***új*** gombot, és adjuk meg a szolgáltató nevét – lehetőség szerint szóköz nélkül, mert ellenkező esetben a parancssorból való indítással bajok lehetnek –, felhasználói nevünket és jelszavunkat, amivel a szolgáltatást elérjük.
A következő párbeszédablakban beállíthatjuk, hogy a kapcsolat csak akkor jöjjön létre, ha egy programunk vagy egy ügyfélgép a hálózatban el szeretné érni az internetet. Ez főleg modemes és ISDN-kapcsolat esetén hasznos lehetőség. Amennyiben úgy gondoljuk, ehelyütt kézzel is meg tudjuk adni a névkiszolgálókat, így nem az internetszolgáltatónk által hozzárendelteket használjuk, hanem a saját beállításunk szerintieket. Továbbá működésre tudjuk bírni a kapcsolathoz a tűzfalat, és be tudjuk állítani, hogy mennyi holtidő után szakítsa meg a gép a kapcsolatot. Ezt az időt érdemes jó nagyra venni, hogy a kapcsolatot ne kelljen állandóan újraépíteni. Az ***IP-részletek*** gomb alatt tudjuk beállítani a kapcsolathoz tartozó állandó IP-címet és hálózati maszkot, ha a szolgáltató nyújtott ilyen lehetőséget a szolgáltatáshoz. Ha ezeket az adatokat megadtuk, végeztünk is az ADSL-kapcsolat beállításaival.

Az utolsó lépések

Következő lépésként létre kell hoznunk az első felhasználót, akit a későbbiekben a felhasználók csoportokba rendezésekor és felvételekor esetleg törölni fogunk. Az adatokat értelemszerűen töltsük ki, a felhasználók és felhasználói csoportok beállításairól a későbbiekben még úgyis szólunk.
A telepítés utolsó lépéseként beállíthatjuk a rendszer által felismert eszközeinket, például a videokártyát, a monitort, a nyomtatókat, a hangkártyát és a tévékártyát. Az utóbbi kettőre gépünk kiszolgálói szerepéből kifolyólag semmi szükségünk nem lesz, úgyhogy akár el is felejthetjük őket.
A videokártya és a monitor beállítása akkor szükséges, ha grafikus rendszert is telepítettünk, szöveges felület használatánál ezek a beállítások elhagyhatók. Amennyiben a géphez nyomtatót is csatolunk, akár már most be is állíthatjuk, bár a nyomtatók hálózati elérhetőségének a beállításakor erről a részről is bővebben fogunk értekezni. (Nyomtatók alapbeállításával SuSE rendszer alatt 2003. novemberi számunk foglalkozott bővebben.)
Ha ezzel a lépéssel is végeztünk, még pár kattintás, és a rendszer tájékoztat bennünket a telepítés befejezéséről és arról, hogy a létrehozott vagy a rendszergazdai jogosultsággal belépve hetünk a rendszerbe, és birtokba is vehetjük azt. Ha beléptünk, még egyszer indítsuk újra a rendszert, így véglegesítve az összes beállítást. Újrainduláskor a rendszer elindítja a beállított szolgáltatásokat, például felépíti a hálózati kapcsolatokat.
Újraindítás után rendszergazdaként bejelentkezve az `ipconfig` paranccsal megnézhetjük, hogy a rendszerben elindultak-e a hálózati csatolók, illetve a megfelelő értékeket vették-e fel. A hálózati kártyák az `ethX` megnevezés alatt, a PPP-kapcsolatok, vagyis a modemes, ISDN-, ADSL-kapcsolatok pedig a `pppX` elnevezés alatt találhatóak. Amennyiben további állításokat szeretnénk végrehajtani a hálózati csatolókon, ezt szöveges terminál esetén a `yast` parancs futtatásával indítható alkalmazásban megtehetjük. A YaST a SuSE rendszerjellemzőnek beállítására szolgáló alkalmazás, amelyet egyfelől részleteiben már használtunk a telepítés alatt, másfelől a későbbiekben is előszeretettel fogjuk alkalmazni.
A hálózati kapcsolattartást a `ping` paranccsal ellenőrizhetjük, de a telepített `links` böngészőt indítva akár weboldalakat is nézegethetünk. Egy SSH-ügyféllel a kiszolgáló távoli elérhetőségét is kipróbálhatjuk, akár windowsos, akár linuxos gépről. Egy böngésző segítségével pedig a kiszolgáló 10000-es kapujára csatlakozva elérhetjük a Webmint.
Ezzel végeztünk is a telepítéssel – van egy elérhető, működő linuxos kiszolgálónk, amit a későbbiekben alapként használunk majd a különböző szolgáltatások telepítéséhez. Próbálgassuk a rendszert, ismerkedjünk a YaST beállítási lehetőségeivel, nézegessük a Webmin rendszer beállításait, és ne féljünk – ha idáig eljutottunk, nagy bajt már nem tudunk okozni a rendszernek. Ha mégis, hát gyakorlásképpen újratelepíthetjük.

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

KAPCSOLÓDÓ CÍM

➲ http://www.suselinux.hu

Debian otthonra (1. rész)

Vallási kérdés vagy makacsság?

Sokszor előfordul, hogy az ember rászánja magát. Igen, végre úgy rendeztem az életem, hogy egy-két szabad délutánt tudtam keríteni, kipróbálom már ezt a Linuxizét az otthoni gépemen. Ezek után már csak néhány alapvető kérdést kell eldönteni, például hogy milyen terjesztést használjak. Ebben a kérdésben rendkívül sok véleményt lehet hallani: sokan az RPM-rendszerek mellett kardoskodnak, és vannak, akik a `deb`-családot részesítik előnyben. Akadnak olyanok, akik a könnyű telepíthetőséget kedvelik, és vannak, akik a kényelmes frissíthetőséget. Előfordulnak olyanok is, akiknek mindegy, hogy egyhetes vagy féléves programcsomagokkal dolgoznak, csak minden szépen működjön, és bizony vannak a türelmetlenek, akik a Mozillának is a legújabb változatát akarják használni, akkor is, ha még nem üzembiztos. És akkor még nem is beszéltünk a „melyik program milyen terjesztések alatt működik rendesen” kérdéskörről.

Melyiket válasszam?

Most közelítsük meg a kérdést a másik oldalról. Mi vehet rá egy kósza lelket, hogy Debiant használjon otthon? Még pontosabban, hogy Sarge-ot, azaz a Debian új, gyorsan frissülő és gyakran változó terjesztését? Kényelmi szempontok? Aligha. Az angol számítástechnikai nyelv gyakorlása? Van benne valami. Vágy az ismeretlen megismerésére? Nna, ez már jobban hangzik. Ha az ember egy kiszolgálót üzemeltet, előbb-utóbb kikristályosodik benne a vágy, hogy megismerje a rendszert. Erre pedig a legjobb környezet az otthoni gépünk, amit úgy kapunk darabokra, ahogy jólesik. Az indokok között szerepel még a kihívás is! Ha valaki azt mondja, hogy ő egy délután alatt felrak egy Sarge-ot egy laptopra, amin zenét hallgat, a böngészőkben flash-sel és Javával készült oldalakat nézeget, ráadásul DVD-filmeket is lejátszik, nos, az ilyen embernek nagyon sok barátja van, akik rendszeresen látogatják, általában egy-egy rekesz sör és egy szűz gép kíséretében.

Nem telepítünk, szerzünk…

A lényeg tehát, hogy otthon elsősorban tanulási, kísérletezési céllal ajánlott Sarge-ot használni. Ebben az írásban pedig igyekszem egy-két olyan célt felállítani, amelyek elérésével egyre otthonosabban mozgunk majd a rendszerben. Mire is lesz szükségünk? Egy gépre, amin van pár giga szabad hely, rengeteg időre, és lehetőleg egy erősebb internetkapcsolatra – ugyanis a folytonosan változó csomagok miatt szinte minden egyes feladatnál folyamatosan kell a csomagokat letöltögetni. Most egy elegáns mozdulattal átugrom a telepítést. Egyrészt azért, mert erről már több szó esett, másrészt ha nem vagy biztos a dolgodban, akkor a példám követését ajánlom: keress valakit, aki éppen ráér, és már gyakorlott Debian-telepítő, kérjed meg, hogy húzzon fel rá egy alaprendszert. A grafikus felületet még kuncsorogd ki, ha tudod. Ha viszont laptopod van, amihez „gyári” Windowst kaptál, figyelj rá, hogy újabban borzalmas módon igyekszenek rávenni más rendszerek kerülésére: az új gépekhez egy olyan „helyreállítólemezt” adnak csupán, amelyikről nem tudsz telepíteni, csupán egyetlen feladatra képes: törölni az egész lemezt (az összes adatot, az összes lemezrészt, mindent), majd az egészet leformázza egyetlen nagy NTFS-lemezrésznek, és rámásol előtelepített Windowst. Ebben az esetben a Linux telepítése előtt egy ügyes lemezméretezővel (például PartitionMagic) kell helyet készíteni az új rendszer számára, és tudd, hogy te már nem akarsz többé „helyreállítani”. Eljutsz tehát valamilyen módszerrel abba az állapotba, hogy van egy valamilyen Debianod, hálózati kapcsolattal. Nagyszerű! Akkor most gyorsan tegyük tönkre! Rendszergazdaként (na jó, a cikk többi részére végig igaz, hogy a rendszerbeállításokat rendszergazdaként kell elvégezni, így ezt többször nem írom le) szerkeszd a */etc/apt/sources.list* fájlt, hogy szerepeljen benne az alábbi két sor:

```
deb ftp://ftp.hu.debian.org/debian/
➥sarge main non-free contrib
deb ftp://ftp.hu.debian.org/debian-non-US/
➥sarge/non-US main  non-free contrib
```

Természetesen más gépet is használhatsz, például ha házon belül is van egy tükör vagy egy apt-proxy. Ezek után kiadsz egy `apt-get update`, majd egy `apt-get upgrade` parancsot, és máris kényelmesen hátradőlhetsz, amíg az a másfélszáz megányi csomag átcsorog hozzád. Mondanom sem kell, ha a vállalati netkapcsolat erősebb, akkor ezt odabent tegyük meg, lehetőleg csúcsidőben, és ártatlan kerek szemekkel nézzünk, amikor tombolva ront be hozzánk a hálózatgazda. Amikor a gép mindent letöltött, elkezdődhet a csomagok telepítése. Ha csak letölteni akarunk, akkor megadhatjuk az `upgrade` mellé a `-dy` kapcsolókat, majd este elindítjuk újra az `upgrade`-t. Telepítés közben nagyon sok érdekeset olvasgathat az emberfia, leginkább olyan dolgokat, amik felhívják a figyelmet arra, hogy mennyire fontos változás történt, az ENTER után pedig nem is emlékszünk majd rá. Nem baj. Egy-két nap múlva már saját magunknak is tudunk rendszert telepíteni.

Néhány gondolat a csomagfrissítések kapcsán

Ha tényleg egy új rendszerről van szó, akkor semmilyen komolyabb egyedi beállítás nincs a gépen (kivéve a netkapcsolat, a Lilo, valamint az X beállításait). Tehát ha a csomagfrissítés kapcsán megkérdezi a gép: „A csudatudja csomag új változata már támogatja a krikszkraksz biztonsági rendszert, de a régi csomagok nem, akarod-e, hogy az új csomag hasz-

nálja az új kiegészítést (de vigyázz, mert a régi rendszerekkel ez nem működik)?" vagy „Az új változat több beállítást használ, felülírjam-e a beállításfájlt?" és alapértelmezett válaszként a „Nem" szerepel, nyugodtan válaszolj „Igen"-nel. A másik fontos dolog a már említett Lilo (ha épp nem Grubot használsz). A rendszeresen visszatérő „Töröljem a korábbi lilo.conf fájlt és készítsek újat?" válaszra határozottan mondj nemet.
Tegyük fel, hogy eljutottál a frissítés végére (esetleg közben falhoz vágtad az újságot, rosszabb esetben a gépet, majd újrakezdted a telepítést, megsétáltattad a kutyát stb.). Hiszed te. Nézzük meg, hogy mit mond erről a kérdésről az `aptitude`. Erről a programról korábban, a Linuxvilág 2002. márciusi számában már írtam (írásunk megtalálható a weboldalunkon is). Ha még nincs a gépeden, telepítsd (`apt-get install aptitude`), majd indítsd el. Ez kétélű fegyver, ugyanis egyik oldalról remekül végignézi, hogy mihez mi hiányzik, másik oldalról viszont néha törölni akar olyan csomagokat is, amelyek azért szükségesek lehetnek. Sőt előfordulhat, hogy telepíteni akarsz egy csomagot, és az `aptitude` se szó, se beszéd, leszedi a csomag régebbi változatát, ráadásul az összes olyan programot, aminek nem jó az új változata, mondjuk az `xfree86-common`-t.

Szóval `aptitude`-ban `g`. Itt megnézheted, hogy mennyire színes a helyzet. Ha vannak zöldek, meg rózsaszínek, akkor az `aptitude` dolgozni akar. Mielőtt elindítod, érdemes végigfutni a listát, sőt visszalépni a fanézetbe (`g`), majd megnézni, hogy van-e törött (broken) csomagunk (hibás függőségekkel rendelkező csomag). Ehhez a legegyszerűbb, ha a ***Limit*** ablakába (`l`) egy `~b`-t írsz, majd ENTER-t nyomsz. Ha egy üres ablakot kapsz, akkor nincs törött csomag. Szóval `g`, majd újra `g`, és már megy is a letöltés, majd a frissítés. Remélem, mondanom sem kell, hogy a csomagfrissítést a karakteres konzolon (CTRL+ALT+F1) érdemes végezni, nehogy a frissítés kiölje alólad az ablakkezelőt. A függőségekkel egyébként jól el lehet játszani, hallottam olyan rendszergazdáról, aki az új munkatársat azzal tette próbára, hogy jól összebarmolta a csomagokat, majd egy „fésüld ki!" felszólítással otthagyta áldozatát.

Indulás!

Több kör futása után tehát eljutottál valamilyen elfogadható állapotba. Ekkor már nincs más dolgod, mint kipróbálni, hogy a rendszer elindul-e a jelenlegi állapotában is. Nézd meg a ***/etc/lilo.conf*** fájlt, hogy benne van-e minden, ami kell (például ha van a gépen, akkor a másik rendszert is el tudod-e indítani), majd add ki a `lilo` parancsot. Ha hiba nélkül lefut, indítsd újra a gépet. Ez az első olyan pillanat, amikor több tudományág egyszerre működik: nevezetesen az informatika és a teológia. Ha sikerrel jártál, akkor visszakapod a bejelentkezési sort, az ég kegyeltjei pedig a grafikus felületet is.
Hogy mondod? Mikor fogunk játszani? Vicces. Mégis mit csinálunk szerinted már napok óta? Mi az, hogy nem érte meg egy darab konzolért ez a felhajtás? Csak most jön a java! Pontosítok, Javával majd csak később foglalkozunk, előbb tűzzünk ki kisebb célt magunk elé.

Rendszermagológia zanzásítva

Most, hogy reményeink szerint minden csomag friss (ami egyáltalán a rendszeren van), gondolkozzunk el, hogy még most kitaláljuk, milyen rendszermagot akarunk használni.
Itt is több lehetőség biztosított. Egyrészt választhatunk a 2.4-es és a 2.6-os sorozat között, másrészt választhatunk az előre lefordított, valamint a „csináld magad" magok közül. Járjuk körül mindkét kérdést. A 2.4-es mellett szól, hogy remekül működik nagyon sok helyen, hosszú ideje folyamatosan javítgatják, mondhatjuk, üzembiztos. A 2.6-os viszont sokkal gyorsabb! A 2.4-es támogatja a `devfs` rendszert, aminek a lényege, hogy nem kell mindenféle idétlen eszközfájlokat karbantartanunk a ***/dev*** alatt, ezt a `devfsd` kezeli. A 2.6-os meg egészen új rendszerrel dolgozik, sőt ha `devfs`-támogatással akarjuk használni, a legtöbbször vérig is sértődik, és már rendszerinduláskor egy „nincs konzoleszköz" hibaüzenettel leáll. Emellett sok minden csak most kezd kialakulni a 2.6-osban. Összefoglalva tehát: a 2.6-os akkor jöhet szóba, ha a sebesség nagyon-nagyon fontos. Egyébként a 2.4-est használd. Személy szerint akkor tettem le a 2.6-os használatáról (ez még a test7-es idején volt), amikor két nap küzdelem után még mindig élettelenül fityegett a billentyűzettől kicsit jobbra az USB-s egerem (a maga büszke kis Microsoft feliratával).
Hogy magadnak fordíts-e vagy sem? Nos, ez megint rajtad áll. A következőt javaslom: egyelőre ne fordíts, használd az előfordítottat, majd egy másik változatszámúval kísérletezz (a cikk írásakor a 2.4.22-es volt elérhető előfordítottként), így nem kavarodnak össze a modulok. Majd amikor az előfordított működési szintjét a saját fordításoddal eléred, akkorra már tudni fogod, hogy melyiket válaszd. Tehát első körben használjuk az előfordítottat. Ehhez az `aptitude`-ban keresd ki a neked tetszőt (ha például egy Pentium III alapú géped van, akkor a `kernel-image-2.4.22-1-686` csomagra lesz szükséged), majd telepítsd azt. Ha azt szeretnéd, hogy a rendszer mindig önműködően telepítse a legfrissebb 2.4-es előrefordított magot, akkor a `kernel-image-2.4-686` csomagot telepítsd (ez mindig kikeresi a legfrissebbet, és azt használja). Értelemszerűen többprocesszoros (vagy duplasínes processzorral működő) Intel alapú gépeken az `smp` végűt, AMD-hez a `k6` vagy a `k7` végűt használd.
Itt is a szabványos négylépcsős folyamatot kell használni: letöltöd, telepíted, imádkozol, újraindítasz. Arra figyelj, hogy a régi magot is el tudd indítani (legyen benne a `/etc/lilo.conf`-ban, valamint a `lilo` parancs ki is írja, hogy beállította).
A rendszer jelen állapotában lényegében semmivel nem tud többet, mint amikor megkaptad, viszont remek alapot teremt a továbbhaladáshoz. A következőkben rápillantunk egy kicsit a grafikus felületre, majd a hangrendszerre. Ha ez megvan, akkor már tényleg szabad az út egy kis Flash-lejátszás felé, és később akár még a Javát is célul tűzhetjük ki! Ezek után ha megkérdezik tőled, hogy otthonra milyen Linuxot ajánlasz, mit mondasz? Igen, helyes a válasz, valóban a Knoppixot!

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

Az XML és a Perl (4. rész)

Az XML-értelmezés DOMináns módszere

Az XML-értelmezés tekintetében a legnépszerűbb módszerek közé tartozik a W3C által megalkotott Document Object Model (DOM). A DOM-értelmező az XML dokumentumot egy objektummodellé fordítja le – ezzel teszi lehetővé a program számára a véletlen elérést és módosítást a dokumentumban. Az objektummodell azt jelenti, hogy az értelmező egy objektumot hoz létre, az úgynevezett dokumentumobjektumot, amely a dokumentumot jelképezi. Mint minden objektumnak, ennek is vannak tulajdonságai, ezek a dokumentum alkotórészeit jelképezik; illetve elemfüggvényei, amelyekkel az utóbbiakat lehet elérni és módosítani. Minden alkotórész-tulajdonság önmagában is egy objektum, és további objektumok lehetnek a tulajdonságai között. Az alkotórész-objektumok felépítése megfelel a dokumentum felépítésének.
A DOM eredetileg egy API-ként fogant meg a HTML-be ágyazott parancsállományok dokumentumeléréshez és -módosításhoz való támogatására, azzal a céllal, hogy a webböngészőben „dinamikus tartalom” jelenhessen meg. Mint az később kiderült, a DOM kiváló módszere az XML-dokumentumok modellezésének. Ez vezetett a szabványos DOM-felület megszületéséhez, amit a W3C ajánlás formájában fogalmazott meg. Majdnem az összes XML-értelmezőnek van DOM-felülete, mi több, a Microsoft XML-feldolgozó eszközei kizárólag DOM-felületet kínálnak. A DOM lehetővé teszi a dokumentumok írását és olvasását, illetve teljesen új dokumentumok létrehozására is használható. A W3C által megfogalmazott DOM tulajdonképpen nyelvfüggetlen felületek gyűjteménye. A DOM megvalósítása egy adott programozási nyelven egy sor nyelvi kötöttséget von maga után; ezek határozzák meg, hogyan lehet az említett felületeket az adott nyelven használni. Például Javában szabványos módon érhetjük el az objektumok tulajdonságait, míg a Perlnek erre nincsenek eszközei. Ebből következik, hogy Javában a szokásos módon érhetjük el a tulajdonságokat, míg Perlben elemfüggvényeken keresztül kapjuk meg és módosíthatjuk őket.
A jelenleg biztonsággal használható W3C DOM-ajánlás, az úgynevezett DOM Level 1 (első szint). A most bemutatásra kerülő Perl DOM-megvalósítás – ***Enno Derksen*** `XML::DOM` modulja – csak az első szintet támogatja, miként a legtöbb kereskedelmi termék is. Az `XML::DOM` tartalmaz néhány, a W3C-ajánlásban nem szereplő lehetőséget is (terjedelmi okokból azonban most nem mutatom be őket). Megjelent a második szint is, amely már szűrőket is tartalmaz és kényelmesebb bejárást tesz lehetővé, ám az ezt megvalósító értelmezők még fejlesztés alatt állnak. Ha bővebb útmutatásra lenne szükséged az `XML::GDOME` modulról, a ➲ http://tjmather.com/xml-gdome/ oldalon megtalálod.

Csomópontok, csomópontlisták, nevesített csomóponttérképek

A DOM-ban minden objektum három osztályból származtatható: `Node` (csomópont), `NodeList` (csomópontlista) és `NamedNodeMap` (nevesített csomóponttérkép). Egy ***csomópontlista*** egy ***csomópontok***-ból álló tömbnek fogható fel, azaz egy ***csomópontok***-ból álló rendezett listának, amelyben az elemekre számokkal hivatkozhatunk. Például ***csomópontlist***-ád lehet egy adott elem gyermekeiből. A ***nevesített csomóponttérkép*** pedig az asszociatív tömbhöz hasonlítható, vagyis nem más, mint egy ***csomópontok***-ból álló rendezetlen lista, amelyben az elemekre névvel hivatkozhatunk, például ***nevesített csomóponttérkép***-ed lehet egy adott elem tulajdonságaiból (attributum).
A dokumentum minden egyes alkotóeleme egy csomópont-osztályból származó objektumként jelenik meg. A `Document Node` (dokumentum-csomópont) jelenti az egész dokumentumot. Az `Element Node` (elemcsomópont) egy elemet jelképez, egy `Attr Node` (tulajdonság-csomópont) szimbolizál egy tulajdonságot, és a `Text Node` (szövegcsomópont) jelképez egy karakterláncot. Léteznek ***csomópont***-ok olyan dolgokra is, mint a megjegyzések vagy a feldolgozói utasítások. A DOM nagyon jól kihasználja az objektumközpontú programozás lehetőségeit; például a ***csomópont*** összes elemfüggvénye használható egy elemobjektumon, és egy ***csomópontlista*** elem- és szövegobjektumokat is tartalmazhat. Ha egy kicsit összezavarnak az objektumközpontú programozás fogalmai, érdemes egy pillantást vetni a perlboot(1) súgóoldalra.
Egy valóságos példa remélhetőleg átláthatóbbá teszi eddig esetleg kissé homályos benyomásaidat a DOM-ról. Nézzük példadokumentumunkat, melyet az előző két részben is használtunk (az 54. CD Magazin/XML_Perl könyvtárában található). Az utóbbinak a DOM-felépítése egy ***dokumentum-csomópont***-ból fog állni, amelynek egyetlen gyermeke a `dokumentum` nevű ***elemcsomópont***. Amennyiben példadokumentumunk a `<dokumentum>` elemen kívül megjegyzéseket vagy feldolgozói utasításokat tartalmazna, a ***dokumentum-csomópont***-nak lennének ***megjegyzéscsomópont***-, illetve ***feldolgozói utasítás csomópont***-gyermekei. Mivel egy jól formázott XML-dokumentumnak csak egy gyökéreleme lehet, a ***dokumentum-csomópont***-nak elvileg csak egy ***elemcsomópont***-gyermeke lehet. Vegyük észre, hogy a ***dokumentum-csomópont*** dokumentum mint állomány és a `<dokumentum>` mint elem kifejezések nem fedik egymást.
A gyökérelemnek, amely a ***dokumentum-csomópont*** egyetlen ***elemcsomópont***-ja, két gyermeke van. Mind a kettő ***elemcsomópont***, és mindkettő neve `fejezet`. Ezeknek az ***elemcsomópont***-oknak az első tulajdonságai között foglal helyet egy ***tulajdonság-csomópont***, ennek a neve `cim`. Az utóbbinak létezik egy ***szövegcsomópont***-gyermeke, amelynek az értéke `elso`, Továbbá egy `bekezdes` és két másik fejezet ***elemcsomópont***-ja is van.

A fa bejárása

Minden ***csomópont*** objektumnak vannak elemfüggvényei, amelyek segítségével ***csomópont***-ról ***csomópont***-ra bejárhatod a DOM fáját. Egy ***csomópont***-ból megtudhatod, melyek a gyermekei, a szülői és a testvérei. A dokumentum bármely határozott típusáról kaphatsz egy ***csomópontlistá***-t, legyen az a címke neve vagy bármi más. Ez azt jelenti, hogy a DOM segítségével nem kell szigorúan felülről lefelé vizsgálni a dokumentumot, mint az `XML::Parser` fa (Tree) stílusa esetén.
Mint láthatod, a DOM hihetetlenül rugalmas és erőteljes felületet kínál az XML-dokumentumokhoz. A beépített eljárások

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

use strict;

use XML::DOM;
use Text::Wrap;

my ($indlevel,@sectnums);

die "Hasznalat: ".$0." {file}\n" unless @ARGV
    ➥== 1;
my $parser = new XML::DOM::Parser;
my $doc = $parser->parsefile($ARGV[0]) or
    die "Nem tudom ertelmezni a
        ➥dokumentumot.\n";

my $root = $doc->getFirstChild;
die "<dokumentum> nem a gyoker elem\n" unless
    $root->getTagName eq "dokumentum";

$indlevel = -1;
$sectnums[0] = 0;
for my $pass (0..1) {
    my $nl = $root->getChildNodes;
    for my $i (0..$nl->getLength-1) {
      my $p = $nl->item($i);
      if ($p->getNodeType == TEXT_NODE) {
          warn "itt nem lehet szoveg" unless
            $pass or $p->getData =~ /^\s*$/;
          next;
      }
      warn $p->getNodeName." nem lehet a
      ➥<dokumentum>-on kivul" unless
          $pass or ($p->getNodeType=
          ➥=ELEMENT_NODE
          ➥and $p->getTagName eq "fejezet");
      process_sect($p,$pass);
    }
}

###############1. szakasz###############
sub process_sect {
    my ($sectnode,$pass) = @_;
    my ($href,$ind);
    if ($pass == 0) {
      ++$sectnums[++$indlevel];
      $sectnums[$indlevel+1] = 0;
      $href =
      ➥join('.',@sectnums[0..$indlevel]);
      $ind = $indlevel;
      $sectnode->setAttribute('href',$href);
      $sectnode->setAttribute('ind',$ind);
      print ' ' x (4*$ind),$href,'
      ➥',$sectnode->getAttribute('cim'),"\n";

###############2. szakasz###############
    } else {
      $href = $sectnode->getAttribute('href');
      $ind = $sectnode->getAttribute('ind');
      print ' ' x (4*$ind),$href,
      ➥'',$sectnode->getAttribute('cim'),"\n\n";
    }
    my $children = $sectnode->getChildNodes;
    for my $i (0..$children->getLength-1) {
      my $p = $children -> item ($i);
      my $t = $p->getNodeType;
      if ($t == TEXT_NODE) {
          warn "szoveg nem lehet a
              ➥<bekezdes>-en kivul" unless
            $pass or $p->getData =~ /^\s*$/;
          next;
      }
      warn $p->getNodeName." nem lehet
      ➥a <fejezet>-ben" unless
          $t == ELEMENT_NODE;
      if ($p->getTagName eq "fejezet") {
          process_sect($p,$pass);
      } elsif ($p->getTagName eq "bekezdes")
{
          my $p1=$p->getFirstChild;
          if ($p1->getNodeType!=TEXT_NODE) {
            warn $p1->getNodeName." nem
            ➥lehet a <bekezdes>-ben" unless
                $pass;
          } else {
            if ($pass) {
                $_ = $p1->getData;
                tr/\n/ /;
                s/^\s+//;
                s/\s+$//;
                my $indent = ' ' x (4*$ind);
                print
                ➥wrap($indent,$indent,$_),
                ➥"\n\n";
            }
          }
      } else {
          warn '<',$p->getTagName,'>
          ➥nem lehet a <bekezdes>-ben';
      }
    }
    --$indlevel;
}
```

megszabadítanak az állapottartó vagy könyvjelző eljárások írásától, amely könnyen felmerülhet egy folyamalapú, de még egy egyszerűbb faalapú értelmező esetén is. Miért akarna bárki is mást használni a DOM-on kívül az XML-értelmezéshez? Talán a redmondi fiúk mégis megtalálták a minden értelemben tökéletes megoldást?

Az igazság az, hogy a DOM elképesztően nehézsúlyú megoldás XML-tartalmak eléréséhez. Egy XML-dokumentum DOM-fája a dokumentum méretének többszörösét falja fel a memóriából. Egy DOM-fa bejárása nem valami gyors és nagyon sok függvényhívással jár, ami Perlben eléggé erőforrás-igényes. Teljesítmény szempontjából határozottan rossz megoldás CGI esetén DOM-ban gondolkozni, ugyanis használata elfogadhatatlan válaszidőkkel és nagy terheléssel járna a kiszolgálón. Mindazonáltal a DOM alkalmazása gyakran leegyszerűsítheti a programodat, javíthatja az átláthatóságot. Tegyük fel, hogy

van egy alkalmazásod, amelyben a sebesség fontos ugyan, de a dokumentum elhanyagolhatóan ritkán változik a program futtatásának számához képest. Elgondolkodtató lehetőség a kinyert adatok tárolása valamilyen egyszerűbb formában, mondjuk DBM adatbázisban. Az alkalmazásod ekkor az utolsó módosítás dátumát összehasonlíthatja az eredeti XML-dokumentumra és az abból nyert adatbázisra vonatkozóan, és csak akkor szükséges újra értelmezni a dokumentumot, ha a lényegi adatokat tartalmazó adatbázis már nem naprakész. Szokásos esetben csak betölti az adatbázist. Amíg nem dolgozol nagyon nagy mennyiségű adattal, addig ez a módszer sokat javíthat az átlagos futási sebességen.

Eleget beszéltünk, lássuk a kódot!

Ahogy azt megtippelhetted, múlt havi programunkat fogjuk újraírni, az `XML::Parser` fastílusa helyett az `XML::DOM` modult használva. Elég nagyvonalúan kezeljük majd a hibakezelést a dolgok egyszerűsítése érdekében. Lássuk!

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

use strict;

use XML::DOM;
use Text::Wrap;

my ($indlevel,@sectnums);

die "Hasznalat: ".$0." {file}\n"
➥unless @ARGV == 1;
my $parser = new XML::DOM::Parser;
my $doc = $parser->parsefile($ARGV[0]) or
     die "Nem tudom ertelmezni a
          ➥dokumentumot.\n";
```

Mivel az `XML::DOM` az `XML::Parser`-ből származik, ugyanazokat a `parse()` és `parsefile()` eljárásokat használjuk. Ha az értelmezés sikeres volt, a visszatérési érték egy hivatkozás a ***dokumentum-csomópont***-ra.

```
my $root = $doc->getFirstChild;
die "<dokumentum> nem a gyoker elem\n" unless
     $root->getTagName eq "dokumentum";
```

Azt akarjuk, hogy a ***dokumentum-csomópont***-unknak csak egy gyereke legyen, egy ***elemcsomópont*** dokumentum névvel.

```
$indlevel = -1;
$sectnums[0] = 0;
for my $pass (0..1) {
     my $nl = $root->getChildNodes;
     for my $i (0..$nl->getLength-1) {
       my $p = $nl->item($i);
```

Kapunk egy ***csomópontlistá***-t, amely gyökérelemünk gyermekeit tartalmazza. A ***csomópont***-ok a listában nullától vannak számozva. Az egyes gyermekeket a ***csomópontlista*** `item()` elemfüggvényével érjük el. A `getLength` mondja meg, hogy hány gyermek található a listában.

```
   if ($p->getNodeType == TEXT_NODE) {
        warn "itt nem lehet szoveg" unless
        $pass or $p->getData =~ /^\s*$/;
        next;
   }
```

A `getData()` elemfüggvény a ***szövegcsomópont***-ok sajátja. Más nyelvekben esetleg `getNodeValue()`-ként fordul elő. Lásd az `XML::DOM` POD-ját.

```
    warn $p->getNodeName."
    ➥nem lehet a <dokumentum>-on kivul" unless
         $pass or ($p->getNodeType==ELEMENT_NODE
         ➥and $p->getTagName eq "fejezet");
    process_sect($p,$pass);
     }
}
```

A `fejezet` elemek feldolgozását a `process_sect()` függvényre bízzuk. (Lásd a lista 1. szakaszát.)
Itt az ***elemcsomópont*** `getAttribute()` és `setAttribute()` elemfüggvényeit használtuk. Ezek karakterláncokat várnak el és adnak vissza. A ***tulajdonság-csomópont*** `getAttributeNode()` és `setAttributeNode()` függvényeivel könnyű őket összekeverni, úgyhogy különös figyelemmel járj el a használatukkor. (Lásd a lista 2. szakaszát.)
A `getFirstChild()` pontosan azt teszi, amit a neve is sugall. Itt mi vakon bízunk abban, hogy `<bekezdes>` elemünk egyetlen gyermeke egy ***szövegcsomópont***.

```
       if ($p1->getNodeType!=TEXT_NODE) {
     warn $p1->getNodeName." nem lehet a
<bekezdes>-ben" unless
          $pass;
       } else {
     if ($pass) {
          $_ = $p1->getData;
          tr/\n/ /;
          s/^\s+//;
          s/\s+$//;
          my $indent = ' ' x (4*$ind);
          print wrap($indent,$indent,$_),"\n\n";
     }
       }
   } else {
       warn '<',$p->getTagName,'>
                ➥nem lehet a <bekezdes>-ben';
   }
     }
     --$indlevel;
}
```

Végszó

Remélem, sok újat és hasznosat sikerült megmutatnom ebben a négyrészes sorozatban. Érdemes legalább egyszer kipróbálni az XML-t, mert sok feladatot általánosítani lehet vele. Ugyanakkor tudni kell, hogy hol lehet használni és éppen melyik értelmezőt, mert a segítségével könnyen írhatunk memóriafalót is. Tanuld meg és használd, csak nyerhetsz vele.

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.

Gépköltöztetések és összeomlás-kezelés

Minden kiszolgáló egyedi, következésképpen mindegyikhez egyedi katasztrófatervvel kell rendelkezni. Készítsük el a saját tervünket, és alkossunk néhány olyan szabályt, amelyekkel rendbehozhatjuk az összeomlásokat.

A rendszerfelügyelet világában, bizony, elég sokat számít a tapasztalat. A rendszergazdákat riogató hibás alkatrészek, pusztító programhibák és biztonsági betörések láttán egyre valószínűbb, hogy érdemes valamilyen hatékony mentési stratégiát, biztonsági rendszabályokat és visszaállítási tervet összeállítani. A kérdés nem is az, hogy bekövetkezik-e katasztrófa a gépünkön, hanem sokkal inkább az, hogy mikor és milyen formában fog megtörténni.
Mindezt 2003 augusztusának közepén írom, kevesebb mint egy héttel azután, hogy a saját kiszolgálómat (***lerner.co.il***) új virtuális helyre költöztettem. A sorok nagy része az elsötétült New Yorkban íródott, ahol néhány órát akartam eltölteni egy üzleti megbeszélésen – végül első kézből tapasztalhattam meg, milyen is egy nagyarányú technológiai katasztrófa. Ó igen, és amikor éppen nem a kiszolgálót vittem át vagy ücsörögtem a sötétben, a hét nagy részében internetkapcsolat nélkül maradtam, minthogy éppen az új lakásomba költöztem be Chicagóban. Ezért azután ebben a hónapban egy kicsit szüneteltetjük a Bricolage-ról és az egyéb tartalomkezelőkről szóló sorozatunkat, helyette megnézzük, hogyan költöztessük át kiszolgálóinkat, és miként készítsük el weboldalaink, adatbázisrendszereink katasztrófaterveit. Természetesen minden webhely és kiszolgáló más és más, tehát rászolgálnak a megkülönböztető figyelemre, vagyis a lehető legjobb tervet készítsük el hozzájuk. Némi előretekintéssel azt mondhatom, hogy nem is olyan nehéz kiszolgálónkat egyik helyről a másikra átvinni, az alkatrészhibákat vagy programösszeomlásokat kezelni, illetve elkerülni az olyan nagyszabású katasztrófát, mint amilyet az Egyesült Államok északkeleti része megtapasztalhatott ezen a nyáron.

Kiszolgálóáthelyezés

Az elmúlt néhány évben párszor át kellett helyeznem a kiszolgálómat, és minden költözés olajozottabban zajlott, mint az előző. Az igazat megvallva egy új gépre történő áttérésnek nem kellene nehéznek vagy fáradságosnak lennie, de mindenképpen körültekintően kell megtervezni. Minden lépést csak olyan módon szabad megtenni, hogy közben azt feltételezzük, hogy arra a pontra esetleg még vissza kell térnünk.
A lehető legegyszerűbb kiszolgálófajta, amit egy másik gépre áttehetünk, a statikus weboldalakból álló és csupán CGI programokat használó webhely. Ilyen esetben csak néhány kérdést kell feltennünk magunknak:

- Tartalmazzák-e az Apache-beállítások felhasznált moduljainkat? Ha sokat használjuk a `mod_rewrite` modult, vagy teljesen kiaknáztuk a `mod_speling` (nem elírás, egy l) előnyeit, nem árt, ha kétszer is ellenőrizzük, hogy megvannak-e ezek a modulok. Amennyiben statikusan építettük (compiled) őket a kiszolgálónkba, a `httpd -l` paranccsal kilistázhatjuk őket. Ha dinamikus modul formájában használjuk (DSO-ként), akkor Apache-telepítésünk ***libexec*** alkönyvtárban kell őket keresnünk, ahol az elérhető DSO-fájlok találhatók. Bármely DSO tetszés szerint betölthető, ha az Apache-beállításfájlban kiadjuk a megfelelő `LoadModule` utasítást.
- Melyik felhasználó és csoport neve alatt fut az Apache? A rendszergazdák véleménye gyakran különbözik, ha az kerül szóba, hogy melyik felhasználói és csoportazonosítót kellene használnia az Apache-kiszolgálónak. Sokan az alapértelmezett ***nobody*** felhasználóként futtatják. Mások (mint például én) jobban szeretik, ha az Apache saját felhasználói és csoportnévvel rendelkezik, és szükség esetén új felhasználókat vesznek fel az Apache-csoportba. Megint csak mások az Apache `suexec` képességét használják ki, úgy fordítják le, hogy egy vagy több másik felhasználóként tudjon futni. Bármelyik esetről legyen is szó, győződjünk meg róla, hogy a kiszolgálónk Apache-felhasználói, illetve csoportazonosítói megfelelően be vannak-e állítva az új kiszolgáló */etc/passwd* és */etc/group* állományaiban, illetve az Apache saját beállításfájljaiban.
- Hol van a `DocumentRoot`? Alapértelmezés szerint az Apache azt feltételezi, hogy `DocumentRoot` értéke */usr/local/apache/htdocs*. Ezt az alapértéket a `DocumentRoot` utasítással tudjuk az Apache beállításaiban megváltoztatni, az általunk használt operációs rendszernek vagy terjesztésnek megfelelően. Amennyiben az Apache RPM alapú változatát használjuk, például valamelyik Red Hat stílusú terjesztéshez, a `DocumentRoot` egyaránt lehet a */var/www* alatt vagy egy másik könyvtárban. Ennek semmi hatása nincs az URL-ekre, illetve a programjainkra és dokumentumainkra nézvést, de azért nem árt, ha kétszer is megnézzük, hogy a könyvtár, amibe az állományainkat másoljuk, tényleg a megfelelő hely-e.
- Milyen nyelveken és modulokon alapulnak CGI programjaink? Amennyiben oldalunk CGI programokat is használ, ezek közül legalább egy valószínűleg használ valamilyen külső modult vagy könyvtárat. A ***CGI.pm***, azaz a Perl module for CGI programs (Perl modul CGI programokhoz) több éve része már a Perl-terjesztéseknek, de továbbra is rendszeresen frissítik. Ezért, amennyiben a legfrissebb változat képességeit használjuk, nem árt, ha biztosra megyünk. Ugyanez vonatkozik minden más felhasznált modulra. Az egyik ügyfelem egy régi Perl ***Storable*** modult használt, és végül rájött (nagy nehezen), hogy az új változatra frissítéssel együtt megszűnt a kapcsolattartás az örökös rendszerekkel.

DNS

Minden kiszolgálóáttelepítés sarokköve a DNS-bejegyzések átmentése. Bár az emberek jobban szeretnek neveket használni

(például *www.lerner.co.il*), a hálózati kapcsolatok olyan számalapú IP-címeket használnak, mint például a 69.55.225.93.
A DNS (Domain Name System, vagyis tartománynév-rendszer) feladata, hogy ezeket az emberi neveket számítógépes számokká alakítsa. A kiszolgálóáttelepítés kényes pontja a DNS-bejegyzések átgondolt módosítása.
A gond a DNS-rendszerrel nem is a gép-IP fordítás ténye, hanem az, hogy a DNS-eredmények gyorstárazódnak. Végtére is mi sem szeretnénk, ha minden egyes HTTP-kérés után a DNS-kiszolgálónknak kellene válaszolnia. Az ilyen kérelmek indokolatlanul nagy forgalmat gerjesztenének, és feleslegesen meghosszabbítanák a HTTP-kérelmek kiszolgálását.
Ezért, amikor DNS-kérelmet adunk ki, valójában nem az eredeti, felhatalmazott kiszolgálótól kapjuk a választ, hanem a helyi DNS-kiszolgálótól. Az ugyanis, amennyiben a kért elemet a mostanában kapott eredmények között megtalálja, rögtön visszaadja, anélkül, hogy a fő kiszolgálótól adatot kérne le. Más szavakkal akkor, amikor az `nslookup www.lerner.co.il` DNS-lekérdezést hajt végre ISP-nk DNS-kiszolgálóján. A kiszolgáló vagy a saját gyorstárából ad vissza adatot, vagy a ***lerner.co.il*** tartomány felhatalmazott kiszolgálójához fordul.
Amikor a kiszolgálót az egyik gépről a másikra átvisszük, érdemes kis számra állítanunk a DNS-kiszolgáló TTL (Time To Live, azaz érvényességi idő) értékét, így az azt gyorstárazó DNS-kiszolgálók nem fognak hibás értékeket visszaadni. Úgy találtam, hogy a TTL értékét bőven elegendő 300 másodpercre (öt percre) beállítani. Ha a kiszolgáló teljesen átköltözött, a TTL értékét ismét növelhetjük valamilyen hosszabb időmennyiségre, például hat órára, hogy csökkentsük kiszolgálónk terhelését.
Nézzük meg nagy vonalakban, hogy milyen lépéseket szükséges megtennünk a sikeres áttelepítéshez, amennyiben HTTP-kiszolgálónkat két szolgáltató között költöztetjük át:

- Győződjünk meg róla, hogy új szolgáltatónk DNS-kiszolgálója képes-e (és hajlandó-e) DNS rendszert (előre és vissza) szolgáltatni jelenlegi a IP-címünk és gépneveink számára. Azaz új szolgáltatónk DNS-kiszolgálójának a régi szolgáltatónkhoz kellene irányítania az embereket. A TTL értékét állítsuk öt percre.
- Frissítsük tartományunk ***WHOIS*** bejegyzéseit, jelezve, hogy az új szolgáltatónk a felhatalmazott DNS-kiszolgáló. Akár egy-két napot is igénybe vehet, mire a dolog a teljes DNS-rendszeren keresztülgyűrűzik. Amennyiben új DNS-kiszolgálónk ugyanolyan eredményt ad, mint a régi, az egyetlen módszer, amivel megnézhetjük, hogy működnek-e a dolgok, ha ***WHOIS***, avagy `nslookup -type=ns` *`tartománynév.com`* típusú keresést végzünk.
- A ***WHOIS***-bejegyzések frissítése után kezdjük meg a dolgok átmozgatását. Győződjünk meg róla, hogy az összes programot helyesen állítottuk-e be, az összes modul megvan-e, és hogy frissítettük-e a DNS-kiszolgálót. Amennyiben a DNS-kiszolgáló tartományunk lekéréseire nem válaszol, igen nagy gondban leszünk, amikor a ***WHOIS***-bejegyzések az új kiszolgálóra fognak mutatni.
- Ha minden azonosnak tűnik (ennek megvalósítására jó módszer lehet az `rsync` használata a régi rendszerről az újra való áttéréskor), a DNS-meghatározásokat változtassuk meg úgy, hogy a gépnév a régi helyett az új IP-címre mutasson.

A futtatott kiszolgáló típusától függően esetleg érdemes kikapcsolni a régi HTTP-kiszolgálót, ezzel is csökkentve az átállás okozta zavart. Például ha kikapcsoljuk a régi HTTP-kiszolgálót, még mielőtt átkapcsolnánk a DNS-t, biztosak lehetünk benne, hogy a naplófájlok között nem lesz átfedés, így azokat a ***Webalizer*** vagy az ***Analog*** eszközökkel összefűzhetjük és felhasználhatjuk, ha körül akarunk nézni bennük.
Ezen a ponton az új rendszeren már mindennek megfelelően kell működnie. Nem árt viszont a lehető legtöbb hivatkozást ellenőriznünk, különösen azokat, amelyek CGI programokat, kiszolgálóoldali csatolásokat és nem hagyományos modulokat hívnak meg, vagy amelyeknek szokatlan jogosultságokra van szükségük. Mint mindig, a folyamat során a HTTP-kiszolgáló hibanaplója lesz a legjobb barátunk; ha esetleg a dolgok rosszra fordulnak, a hibanaplóban általában megtalálhatjuk, hogy mi lehet a gond.

Adatbázisok

Természetesen a fentiek során feltételeztük, hogy egy viszonylag egyszerű oldallal dolgozunk. A legtöbb korszerű weboldal azonban ilyen vagy olyan okból kifolyólag relációs adatbázisokat is tartalmaz. Használatuk egyaránt népszerű egységességük és rugalmasságuk, valamint a gyors fejlesztés lehetősége és az általánosan használt paradigmák beépíthetősége folytán, ezeket könnyű használni és a hibáikat elhárítani.
A relációs adatbázisok egy vagy több táblában tárolják az adatokat, amelyeket általában adatbázisba szerveznek. (Igen, némileg zavaró, hogy egyetlen adatbázis-kiszolgáló több adatbázist is tartalmazhat, amelyek mindegyike egy vagy több táblát is tárolhat, de pontosan ez a helyzet.) Ha adatbázisunkat az egyik rendszerről a másikra szeretnénk átvinni, egyaránt át kell juttatnunk az adatbázissémát (táblák, nézetek és függvénymeghatározások), valamint magát az adatot. Természetesen az adatbázis tulajdonosa valamilyen adatbázis-felhasználó (aki általában nem azonos a Unix-felhasználóval), és egyedi jogosultságkészlettel rendelkezik rajta.
Amennyiben honlapunk adatbázist is tartalmaz, ezt is át kell vinnünk a régi rendszerünkről az új alá, ideértve a tulajdonosokkal és a jogosultságokkal kapcsolatos adatokat is. Hogy ezt miképpen kell megtennünk, attól függ, hogy milyen adatbázist használunk, és van-e valamilyen buktatója e folyamatnak.
Az ISAM/MyISAM táblákat (az alapértelmezett és máig legnépszerűbb lehetőség) használó MySQL-adatbázisokat az egyik MySQL rendszerről egyszerűen átmásolhatjuk a másikra. Általában az adatbázissal kapcsolatos valamennyi állományt megtaláljuk a */var/lib/mysql* könyvtárban, az adatbázis nevével azonos könyvtárban. Így, ha a *foo* adatbázist szeretnénk átvinni, másoljuk a */var/lib/mysql/foo* könyvtárat és a benne lévő állományokat az új gépünk */var/lib/mysql/foo* könyvtárába. (Mielőtt ezt megtennénk, ne felejtsük el lezárni az adatbázist.) Indítsuk el az új rendszer kiszolgálóját, és mindennek jól kell működnie.
PostgreSQL alatt, amely az adatbázissémákat és az adatokat alacsony szintű bináris formában tárolja, már korántsem ilyen egyszerűek a dolgok. Amennyiben `tar` vagy `rsync` programmal próbálnánk átmásolni PostgreSQL adatbázisunkat, igen valószínűtlen, hogy működne; és ha mégis, valószínűleg komoly adatkárosodást fog szenvedni, sőt akár a háttéradatbázis-kiszolgálót is lefagyaszthatja. Ehelyett használjuk inkább a `pg_dump` eszközt, amellyel a PostgreSQL adatbázist `CREATE`, `INSERT` és `COPY` parancsok sorozataként, egyszerű szöveges formátumban kimenthetjük. Például:

```
pg_dump -U mydb mydb > /tmp/mydb-dump.txt
```

A `-U mydb` szakasz azt jelzi, hogy a `mydb` adatbázis-felhasználót szeretnénk használni. Ide a megfelelő nevet kell beírni.

A kimentett adatokat a következő paranccsal lehet visszatölteni egy működő adatbázisba:

```
$ createdb -U mydb mydb2
$ psql -U mydb mydb2 < /tmp/mydb-dump.txt
```

A parancsokat követve most két adatbázissal rendelkezünk (***mydb*** és ***mydb2***), amelyeket egyaránt a `mydb` felhasználó birtokol. MySQL alatt igen könnyű az ilyen helyzetek kezelése, hiszen egy beépített elsődleges, illetve másodlagos rendszerrel rendelkezik. Az egyik adatbázis lehet az elsődleges, ez fogadja majd az összes SQL-parancsot és lekérdezést, míg a másik csendben követi, lehetővé téve, hogy összeomlás esetén felváltsa az elsődlegest.

Áramszünet esetén...

Mint korábban említettem, augusztusban New York városában volt „szerencsém” megtapasztalni azt a komoly áramkimaradást, amelynek emberek milliói érezhették hatását az Egyesült Államokban és Kanadában. Ironikus, de a kimaradás körülbelül egy órával az után történt, hogy az egyik potenciális ügyfelünk kiszolgálótelepét meglátogattam, ahol bemutatta, miképpen menti cégük az összes adatukat egy connecticuti távoli csomópontra. (Végtére is mennyi az esélye annak, hogy valami egyszerre hasson New York városban és Connecticutban? Legalábbis erre gondoltam, miközben helyeslően bólogattam egy órával a kimaradás előtt.)

Ha valaki az enyémhez hasonló helyzetben volt, New Yorkon kívül található kiszolgálóit nemigen zavarta a kimaradás. Ráadásul a legtöbb szolgáltatóhely háttérgenerátorral is rendelkezik, amely szükség esetén részben vagy egészben fedezni tudja az épület áramigényét.

De ha kiszolgálóink az irodában állnak, vagy egy egyszerű szünetmentes tápegységre (UPS) bíztuk futásuk állandóságát, a kiszolgáló valószínűleg elérhetetlen lesz egy olyan áramszünet esetén, mint amilyet augusztus közepén megtapasztalhattunk, s amely egyes északkeleti részeken 48 óránál is tovább tartott. Amennyiben kiszolgálónk üzletünk létfontosságú része, komolyan érdemes elgondolkodnunk azon, nem lenne-e érdemes egy társszolgáltatónál elhelyezni.

De időnként még a társszolgáltatónál elhelyezett kiszolgálók is leállhatnak és lekapcsolódhatnak – ezt többéves, saját tapasztalataim alapján állíthatom. Ez azt jelenti, hogyha nagymértékben függünk a kiszolgálótól, rendszeresen mentenünk kell. Továbbá folyamatosan másolnunk kell egy másik fizikai helyen (és lehetőleg egy másik cégnél) elhelyezett kiszolgálóra.

A különbségeknek ezzel vége is – a programbeállítások jobb, ha azonosak maradnak. Amennyiben kiszolgálónkon a HTML-oldalakhoz, sablonokhoz és programkönyvtárakhoz, valamint a CGI programokhoz az `rsync` eszközt használjuk, és hasonlóan automatizáljuk az adatbázismentések áttöltését a második kiszolgálóra, a második kiszolgáló közel azonos másolata lesz az elsőnek, és a megfelelő pillanatban szolgálatba lesz állítható. Ennél is továbbléphetünk és akár egyszerre is használhatjuk a két kiszolgálót. Természetesen ez már jóval nehezebb feladat, mivel megköveteli, hogy vagy egyetlen adatbázis-kiszolgálót használjunk (s így egyetlen helyre bízzuk rá az adatokat), avagy az adatbázisokat sűrűn össze kell hangolnunk. Ennek ellenére ez mindenképpen lehetséges megoldás, különösen azokon a helyeken, amelyek nagy, statikus oldalakat tartalmaznak – ez jól látszik az Akamai sikeréből, amelynek számos redundáns kiszolgálója van szerte a világon. Minél statikusabb egy oldal, annál könnyebb lemásolni és a futását folyamatossá tenni.

Az üzleti adatbázisprogramoknak (például Oracle) még mindig van egy előnyük a PostgreSQL és MySQL rendszerekhez képest. Igaz ugyan, hogy az utóbbi bizonyos mértékben képes elsődleges, másodlagos másolatkészítésre, de az összehangolás egyáltalán nem olyan kifinomult és nem is annyira hibatűrő, mint az Oracle nyújtotta változata. Idővel várhatóan ez a helyzet is megváltozik, ahogy az ilyen megoldások egyre gyakoribbakká válnak.

Összegzés

Új helyre költözni nehéz, kiszolgálónkat új helyre vagy gépre költöztetni nemkülönben. Egy jó áttelepítési tervvel felvértezve, lépésenként haladva és munkánkat minden ponton ellenőrizve (olyan segédeszközökkel, mint az `nslookup`, `dig`, `telnet`, valamint a Perl LWP, vagy a hasonló műveleteket végző `curl` eszközkészlettel szállított HEAD és GET programok) átállásunk zökkenőmentes lehet.

Linux Journal 2003. november, 115. 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 Izraelben, Modi'inben él.

Izzítsuk be a processzorainkat!

Linux-szakácsunk kedvenc versenyautós játékai a kiszámíthatatlantól a szuperrealisztikusig terjednek. Készüljünk fel egy kis gumiégetésre!

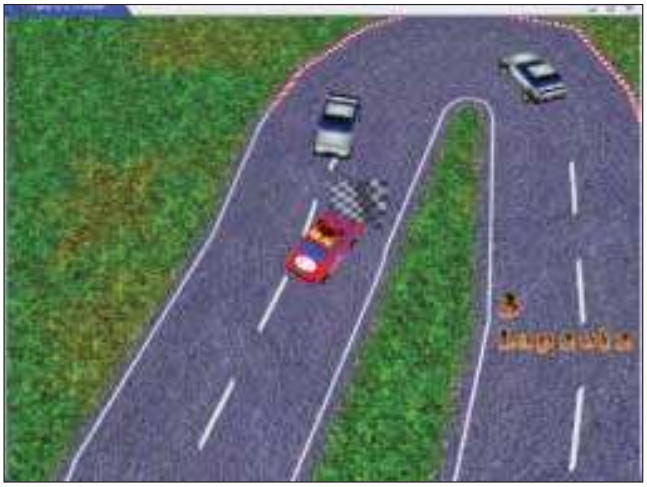
Igazad van, François, pontosan így érzek én is. Bár látom, hogy ennek a számnak a nagyteljesítményű számítások állnak a középpontjában, mégis, amikor nagy teljesítményre gondolok, leginkább a versenyautók jutnak az eszembe. Különös, mon ami, de erős kapcsolat fűzi ezt a két dolgot egymáshoz. Végül is mi feszegeti annyira a számítási teljesítmény korlátait, mint egy jóféle 3D-s szimuláció? Gondoljuk csak meg: a nagy teljesítményű autók versenye így vezet a nagy teljesítményű számításokhoz. Ez aztán a remek, sőt mámorító összefüggés, nem igaz?

Á, épp jókor! Megérkeztek a vendégeink, François! Isten hozott titeket Chez Marcelnél, az ízletes Linux-konyha, a különleges borok és lélegzetelállító autóversenyek házában. Foglaljatok helyet, helyezzétek magatokat kényelembe. Remélem, tetszik a mai díszlet. Versenycsíkokat festettünk minden székre és asztalra a nagy teljesítményű számítások témájának tiszteletére. François! A pincébe, immédiatement! Szükségünk van valamire, ami felrázza az érzékeinket. Egy korábbi pincebeli minőségellenőrző körutam emléke alapján az Ausztráliából származó 1999-es Margaret River Chardonnayt elég izgalmasnak és tüzesnek tartom ahhoz, hogy neki merjünk vágni a távolságoknak. Amíg François-ra várunk, hogy visszatérjen a borral, elmondom, hogy a mai menü minden fogásához egy 3D-gyorsítóval rendelkező videokártyára és a fordításhoz a megfelelő XFree86-meghajtókra lesz szükségünk, köztük a Mesa 3D fejlesztői csomagokra. Itt, az étteremben minden rendszer fel van készítve, de ha az otthoni 3D-gyorsító beállításához útmutatásra lenne szükségetek, lapozzátok fel a Linuxvilág 2003. júniusi számában megjelent „Csaták a számítógép belsejében” című cikkemet, amelyben a közvetlen leképezésről és a kártya teljesítményének a kipróbálásáról is olvashattok.

A Race

Az első autóverseny-szimulációt, amire vissza tudok emlékezni, nem számítógépen valósították meg: egy egyszerű modellautóverseny volt. Habár valódi háromdimenziós élményt nyújtott – hiszen semmi nem lehet a valóságnál háromdimenziósabb, non? –, felülnézetből lehetett látni, tehát egyfajta felülnézeti 3D-s megvalósításról beszélhettünk. Ebben a szellemben készült a ma esti menü első fogása, ***Harry Storbacka*** Race nevű programja is. Ahhoz, hogy a program működésre készen álljon (vagy mi álljunk készen a Race futtatására), választhatjuk a bináris változat letöltését a honlapról, vagy magunk állíthatjuk elő forráskódból. A két megoldás mindegyikét támogatja a program honlapja (➲ http://race.sourceforge.net). Természetesen a bináris állomány kicsomagolása a legegyszerűbb, de ha úgy döntenénk, hogy forráskódból fordítjuk le, akkor rendelkeznünk kell a ***clanlib***, ***xml2*** és ***ode*** fejlesztői könyvtárakkal.
A forráscsomag kibontása után már egyszerűen csak a `make` futtatása van vissza, mint az alább is látható:

```
tar -xzvf race-0.9.0-src.tar.gz
cd race-0.9.0
```



1. kép A játék arra is emlékszik, hol csúsztunk ki

```
make
./race
```

A telepítés nem különösebben elegáns (legalábbis pillanatnyilag). Azt tapasztaltam, hogy egy kicsit még játszadozni kell a `Makefile`-lal (nevezetesen az ***xml2*** könyvtárakra mutató elérési útvonallal), úgyhogy a rendelkezésre álló bináris fájl futtatása kétségtelenül könnyebb. Csomagoljuk ki (`tar -xzvf race-0.9.1-0-static-linux.tar.gz`), váltsunk át a könyvtárra és futtassuk a ***./race-0.9.1-static*** fájlt. A játék azzal indul, hogy néhány beállítást – többek közt a pályát – ki kell választanunk. Kattintgathatunk a ***Continue*** (folytatás) gombra is, amíg a játék el nem indul. Miként már említettem, felülnézetből látjuk a versenyt. Ha nem nyomjuk eléggé a gázpedált, a többi pályán lévő autó nagyobb sebességre próbál meg minket ösztönözni. A játékmenet furcsán realisztikus. Amikor a kerekek megcsúsznak, füst száll fel a gumikról. Az elinduláshoz az A billentyűt (accelerate, azaz gyorsítás) kell nyomnunk, a nyilakkal pedig jobbra, illetve balra kanyarodhatunk. Néhány kisodródás után tapasztaltam meg, hogy milyen apró részletekre is odafigyeltek a játék megírásakor: amikor második alkalommal is elértem ugyanazt a helyet, a csúszás nyomai még mindig az úton voltak. Nagyon hatásos.
A versenyláz akkor kezdi igazán elkapni az embert, amikor (akár virtuálisan is) egy autó kormánykereke mögé ülhetünk – ez megmagyarázza a kedvenc játéktermünkben lévő versenygépek izgalmát és vonzerejét. Linux alatt is számos ilyen típusú szimulátort találhatunk. Néhányuk egészen kiforrott és profi, de – akárcsak a való életben – az autók és a motorjaik állandó fejlesztés alatt állnak, és a határokat feszegetik, ahogy némi többletfordulatszámot próbálnak meg kisajtolni magukból. Ugyanez a helyzet a nyílt forrású fejlesztésekkel is, amelyek közül most bemutatok néhányat.

2. kép Próbálok az úton maradni a T1 Car Racing Simulationnel

3. kép A futurisztikus Canyon Racer

4. kép Szabad a pálya a TORCS-ban

Ígéretes kezdők

Különösen ígéretes program ***Alex Pozgaj*** T1 Car Racing Simulation (t1-crs) nevű alkotása. Írásom megszületésének az idején a játék az alfaállapotnál tartott, mégis igen jó szórakozást nyújtott, annak ellenére, hogy nem volt teljes mértékben játszható. Ha ki szeretnénk próbálni egy körre (például Alex Toyota Suprájával), látogassunk el a T1 honlapjára a ➲ http://t1-crs.sourceforge.net címen. A forráskóddal felszerelkezve a következő lépéseket kell követnünk:

```
tar -xzvf t1-crs-0.1.2a.tar.gz
cd t1-crs-0.1.2a
./configure
make
```

A program még nem rendelkezik telepítővel. A játék elindításához maradjunk a fordítás könyvtárában, és adjuk ki az `src/t1_crs` parancsot. A kurzorbillentyűkkel fordulhatunk jobbra, illetve balra, és ugyancsak a nyilak szolgálnak gáz- és fékpedálként. Ne felejtsünk el indulás előtt sebességet váltani, ellenkező esetben előfordulhat, hogy hátramenetben találjuk magunkat. A billentyűzet Q és A gombjaival válthatunk nagyobb, illetve kisebb sebességfokozatba, és mint tudjuk, ez nélkülözhetetlen is az elinduláshoz.

A mai esti menü következő fogásának szerzői – foobar és judeo – alkotásukat OpenGL Race Game néven emlegetik, én viszont Canyon Racernek fogom hívni, megkülönböztetve a ma még sorra kerülő többi OpenGL versenyjátéktól. A Canyon Racer szintén a fejlesztési szakasznál tart, de ettől függetlenül ezzel is jól szórakozhatunk. A Csillagok háborújából ismert fogatok (pod racer) sajátosságai alapján a játék egy futurisztikus, lebegő járműbe invitál, amivel a szurdok falai között száguldozhatunk. A játékmenet igen gyors, és egy kicsit vad, ahogy a falak között próbál maradni az ember. Balra fent egy térképrészletet láthatunk, ami a közeledő kanyarra hívja fel a figyelmünket. Beismerem, mes amis, nekem épp elég gondot jelentett a falak elkerülése, nemhogy még a térképet is nézzem. A Canyon Racer egy példányának beszerzéséhez látogassatok el a Project Z honlapjára, a ➲ http://projectz.ath.cx/?id=70 címre, és töltsétek le a forráskódot. Ahogy tulajdonképpen minden bemutatásra kerülő játék, ez is egy 3D-gyorsítóval rendelkező videokártya meglétét igényli. A játék lefordításához az OpenGL és SDL (keverő és kép-) programkönyvtárakra van szükségünk. Ha a feltételek adottak, a többi már nem jelent gondot:

```
tar -xjvf racer-0.5.tar.bz2
cd racer-0.5
make
```

Mivel nincs telepítő parancsfájl, a játék a fordítás könyvtárából indítható. Gépeljük be a `./race` parancsot, és már sínen is vagyunk.

Vitathatatlan kedvencem

A kedvenc autóversenyem (és egyben ennek az összeállításnak a legfejlettebb darabja) a TORCS. A TORCS projekt vezetője, ***Eric Espié*** és csapata kiforrott és fejlett technikával létrehozott autóversenyt alkotott, gyönyörű grafikával, fotószerű látványnyal, valós idejű működéssel és rengeteg különböző autóval (az írás idején több mint negyven közül választhatunk).
Ha unatkozni kezdenénk a TORCS-szal, itt az ideje, hogy részesévé váljunk a kalandnak. A program lehetővé teszi, hogy beprogramozzuk a saját autónkat, az ellenfelek robotpilótáit, és a versenypályák nyomvonalát. Ez a játék komoly versenyzők számára készült.
A TORCS egy példányának megszerzéséhez látogassunk el a ➲ http://torcs.sourceforge.net oldalra. A honlapon a forráskód

mellett előre fordított csomagokat is találunk a Red Hathoz, a SuSE-hez, a Mandrake-hez, a Debianhoz és más rendszerekhez (végtére is a program a GPL alá tartozik). A nagy teljesítményű versenyt a végsőkig kihasználni szándékozók számára CVS-letöltések is rendelkezésre állnak.
A forráskódból való fordítás lényegében a szabványos utat követi, azonban számos programozói könyvtár megléte szükségeltetik a 3D-fejlesztéshez (nevezetesen a Mesa és a GLUT), csakúgy, mint a plib. A legegyszerűbb megoldás valamelyik bináris állomány letöltése. Ha ezt választjuk, figyeljünk rá, hogy minden szükséges összetevő rendelkezésre álljon!
Az alap ***TORCS*** és a ***TORCS-data*** csomagok megléte elengedhetetlen. Bár az induláshoz ennyi is elég, azért töltsünk le és telepítsünk néhányat a ***TORCS-robots***, ***TORCS-data-cars*** és ***TORCS-data-tracks-base*** csomagokból is. Ezek révén ellenfeleket, nagyon jó versenyhelyszíneket és a választható autók már említett széles kínálatát nyerjük el.

KAPCSOLÓDÓ CÍMEK

Harry Storbacka Race programja
➲ http://race.sourceforge.net
Az OpenGL Race Game (átkeresztelve Canyon Racer)
➲ http://projectz.ath.cx/?id=70
T1 Car Racing Simulation ➲ http://t1-crs.sourceforge.net
TORCS ➲ http://torcs.sourceforge.net
Marcel borlapja ➲ http://www.marcelgagne.com/wine.html

A TORCS elindításához a `torcs` parancsot kell begépelnünk. Az első, amit látunk, egy egyszerű képernyő, amely egy egyjátékos menetet és beállítási lehetőségeket kínál. Ha türelmetlenek vagyunk, kezdjük rögtön egy egyjátékos versenynyel, viszont később biztosan visszatérünk még néhány beállítás megváltoztatása kedvéért. A TORCS működik billentyűzetről, az egérrel vagy a botkormánnyal, a beállítás pedig lehetőséget ad ezek finomhangolására is. Ugyancsak itt változtathatjuk meg játékosunk nevét, választhatunk autót vagy pályát, dönthetünk a sebességváltó működésének kérdésében (automata vagy kézi) és a többi. Még az alapverseny menüjében is dönthetünk olyan kérdésekben, mint hogy milyen autót szeretnénk vezetni és melyik helyszínen. Én személy szerint a piros Ferrari kormánya mögött szeretek ülni az Alpokban található pályán.
Az idő elszaladt, mes amis, látom magunk előtt lengeni kockás zászlót – elértük a zárórát. A vôtre santé! Bon appétit!

A játékok megtalálhatóak az 54. CD Magazin/Fogado könyvtárában.

Linux Journal 2003. november, 115. 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 (jelenleg is egy könyvön dolgozik).

Átállás Linuxra – intsünk búcsút a Kék Halál képernyőnek!

Marcel Gagné, díjnyertes író tollából új könyv született. Ennek magyar nyelvű kiadása előkészületben van a Kiskapu Kiadónál.
A könyv segít benne, hogy Windows-rendszerről akár néhány óra alatt zökkenőmentesen Linuxra váltsunk. Mire befejezzük a könyvet, a Linuxszal jóformán mindenre képesek leszünk, egyúttal megszabadulunk a Windows futtatásával járó idegeskedéstől, rendszerösszeomlásoktól, biztonsági kockázatoktól és magas költségektől.
A könyv nem a műszaki zseniknek szól, hanem az olyan felhasználóknak íródott, akik dokumentumokat írnak, táblázatokkal dolgoznak, a világhálót látogatják, elektronikus levelezést folytatnak, CD-t hallgatnak, számítógépes játékokkal játszanak – és mindezt egyszerűen Linuxszal szeretnék folytatni, anélkül, hogy különösképpen műszaki szakértőkké kellene válniuk.
Ízelítő a könyv tartalmából:

- Változtassuk windowsos gépünket linuxos rendszerré, amely kevesebb pénzért többet nyújt!
- Böngésszünk a világhálón, küldjünk és fogadjunk elektronikus leveleket, sőt meglévő AOL, MSN vagy Yahoo! azonosítónk használatával küldjünk azonnali üzeneteket!
- Csatlakoztassuk a géphez digitális fényképezőgépünket vagy lapolvasónkat, és fedezzük fel a Gimpet, a Linux hatékony képszerkesztő programját!
- Olvassunk be zenéket, írjunk és játsszunk le CD-lemezeket a Linux hihetetlen mennyiségű, könnyen használható multimédia-eszközeinek segítségével!
- Fedezzük fel a linuxos játékok világát a Solitaire-től a repülőgép-szimulátorokig és azon is túl!

És ezzel még nincs vége: készítsünk dokumentumokat, végezzünk számításokat, hozzunk létre bemutatókat az OpenOffice.org programmal, ezzel az ingyenes linuxos irodai programcsomaggal, amellyel írni és olvasni is tudjuk már meglévő, MS Office-ban létrehozott fájljainkat. Mondjunk búcsút a drága programfrissítéseknek, a fárasztó Microsoft-licencelésnek, a Windows vírusainak és Kék Halál képernyőjének. Köszöntsük a számítástechnikának azt az arcát, amilyet mutatnia kellene: a Linuxot!
Röviden a CD-mellékletről: egy rendszerindító Linux-rendszert, a Knoppixot tartalmazza, amely anélkül bizonyítja a Linux hatékonyságát, egyszerűségét és használhatóságát, hogy meglévő Windows-rendszerünkhöz hozzá kellene nyúlnunk.

(X)

nefu

A nefu egy olyan újabb szolgáltatásfigyelő program hálózati kiszolgálók számára, amely számos jól ismert szolgáltatást tud ellenőrizni, illetve saját parancsfájlok írását teszi lehetővé. Más hasonló programokkal szemben a nefu tudja, mi az a függőség – azaz ha

megmondod neki, hogy közte és a levelezési kiszolgáló között egy útválasztó található, akkor ha az útválasztó elérhetetlenné válik, a nefu elektronikus levélben értesít róla. A levelet neked küldi, nem pedig a levélkiszolgálóra, amely esetleg szintén nem érhető el. Futtatásához szükséges: libresolv, libnsl, libssl, libcrypto, glibc, libdl.

➲ http://rsug.itd.umich.edu/software/nefu

synonym

Meg sem tudom számolni, hányszor kérdezték már meg tőlem, hogy miként lehet átmásolni egy levélkiszolgálón

áthaladó összes – szó szerint az összes bejövő és kimenő – üzenetet fájlba vagy adatbázisba. Nos, a synonym egy olyan Sendmail szűrőprogram, amely pontosan ezt teszi. A Sendmail által feldolgozott minden egyes üzenetet egy felhasználóhoz másolja. Továbbá ***X-Copied-To:*** (átmásolva ide) fejléceket is felvesz, nyilvánvalóvá téve, hogy az üzenet valóban archiválva lett. A futtatásához szükséges: libpthread, libsm, libsmutil, libmilter, sendmail milter és glibc.

➲ http://www.modulo.ro/synonym

JFFNMS

A „just for fun network management system” (JFFNMS, magyarul „a móka kedvéért hálózatkezelési rendszer”) valójában több mint móka. Figyeli a

rendszert, és úgyszólván mindenfajta tevékenységet ábrázol, amely SNMP-vel észlelhető. Az adatokat SQL adatbázisban tárolja. Ezenkívül a `tftp` segítségével biztonsági mentést készít olyan eszközök beállításairól, mint például a Cisco útválasztók vagy a vezeték nélküli hozzáférési pontok. Futtatás: Apache, PHP MySQL-lel vagy PostgreSQL-lel, SQL-kiszolgáló, SNMP, RRDTool, tftp (nem kötelező).

➲ http://jffnms.sourceforge.net

SolarWolf

Íme, egy újabb remek unaloműző. Ha szereted az Arcade stílusú játékokat, ez a program többórás szórakozást biztosít

számodra: tűzlabdákat kerülgetve dobozokat kell gyűjtened. Az animáció és a grafika kitűnő, a hang nemkülönben. Bárcsak lenne benne ágyú is, hogy lőhessek a tűzlabdákra! Működéséhez Python és Pygame modul szükséges.

➲ http://pygame.org/shredwheat/solarwolf

Jabber

Ha valaha használtad már az MSN Messengert, az AIM-ot vagy a Yahoo Messengert, és egy azonnali üzenetküldési rendszert szeretnél felállítani a vállalatod számára, hadd ajánljam a figyelmedbe a Jabbert. A Jabber protokoll valójában sokkal többre képes jogdíjas versenytársainál. Ezen túlmenően a Jabberrel csatlakozhatsz az MSN-hez, az AIM-hoz vagy a Yahoo-hoz, habár ehhez egy bejelentkezési fiók is szükséges ezeken a rendszereken. Futtatásához elengedhetetlen: libcrypto, libdl, libresolv, glibc és libssl (nem kötelező).

➲ http://www.jabber.org

Tkabber

A Jabber-kiszolgáló mellett egy Jabber-ügyfélre is szükség van. A linuxos kínálatot áttekintve arra jutottam, hogy a Tkabber egyike a legkönnyebben

kezelhető alkalmazásoknak, és ez nyújtja a legtöbb szolgáltatást az összes – jogdíjas és szabad program – közül. Előfeltételei: tcl/tk, wish, tcllib, bwidget.

➲ http://tkabber.jabber.ru/en

dos2unix

Bizonyára sokuknak akadt már nehézsége korábbi MS-DOS vagy Windows-rendszerek alatt létrehozott szövegfájlok olvasásával vagy épp továbbszerkesztésével. Ennek az az oka, hogy a soremelés DOS alatt 2 bájtos (CR/LF), míg a Unix-rendszerek alatt ezt egyetlen bájttal oldják meg. A fenti nehézségre jelenthet egyszerű megoldást a `dos2unix` nevű átalakítóprogram, amely a soremelések kicserélgetésével unixos formátumúvá alakítja a szövegfájlt. Ezen túl lehetőség nyílik ASCII és ISO módok használatára is.

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

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

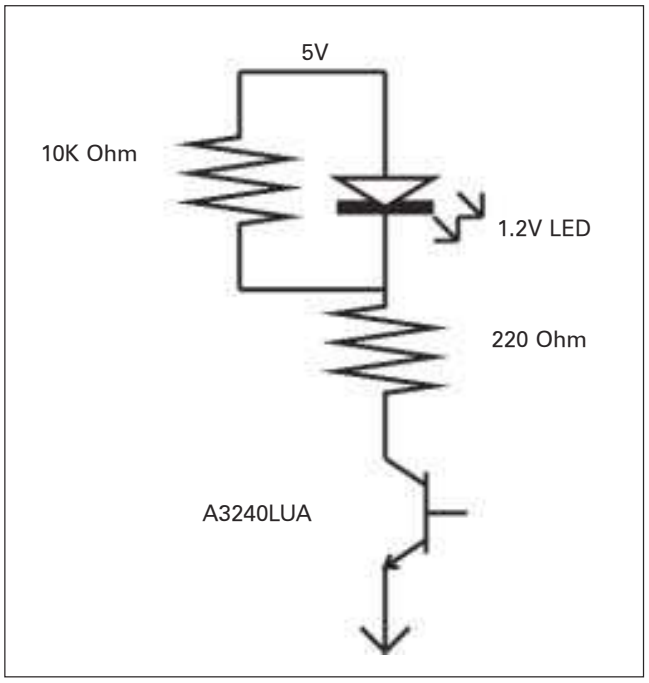
Szennyvízszivattyú a weben, beágyazott Linuxszal

Egy egyszerű áramkörrel bármilyen váltakozó árammal üzemelő elektromos készülékről megállapíthatjuk, hogy be van-e kapcsolva, és az adatokat weben keresztül is elérhetővé tehetjük.

Feleségemmel 1996 nyarán vásároltuk meg a házunkat. 1997 kora tavaszán az alagsor tele lett vízzel, rá egy évre a jelenség megismétlődött. 2001 tavaszán az árvíz újfent ismét elöntötte az alagsort. Ekkor egy kissé kezdtük unni a dolgot, és elhatároztuk, hogy felszerelünk egy szennyvízszivattyút. 2003-ban, amikor rápillantottunk arra a több mint egy méter magas hókupacra, amely csakis arra várt, hogy elolvadva a mi pincénkbe szivárogjon be, eszünkbe jutott, hogy végül is a szivattyú felszerelésére azóta sem került sor. Ekkor keményen elhatároztuk, hogy cselekedni fogunk, és meg is tettük. Mindössze egy hétig kellett fúrókalapáccsal ügyeskedni a pincében, és máris körbecsövezett pincével, valamint egy csinos kis szivattyúval dicsekedhettünk, amely akkumulátoros tápot és tetszetős borítást is kapott. Mivel kételkedtünk abban, hogy kedves kis szivattyúnk képes-e megküzdeni a hólével, kissé rögeszmévé válva egyre többet forogtak a gondolataink a betörő víz körül. Jómagam tízpercenként ellenőriztem a vízszintet. Éjjelente azért keltem fel, hogy a vizet nézzem. A munkahelyemről rendszeresen hazatelefonáltam, hogy helyzetjelentést kérjek – szóval, az egész kezdett komédiába illő lenni.
Az a szerkezet, amit a saját nyugalmam érdekében állítottam üzembe, egyre jobban felemésztett. Végül úgy döntöttem, egy olyan megoldás kell, amivel távolról is figyelni tudom a szivattyú működését. Meghatároztam házi tervezetem fő céljait:

1. ne égjen le a ház;
2. a szivattyú működését a figyelőeszköz nem gátolhatja meg;
3. tanulni akarok valami újat.

Első és második számú célommal összhangban úgy határoztam, hogy a szivattyú áramellátásával semmit nem szabad sorba kötnöm. Nyilvánvaló volt, hogy a saját tervezésű áramkörömön – biztonsági okokból, például a jelek feldolgozását végző processzor leválasztásának és védelmének nehézségei miatt – inkább ne folydogáljanak 10 amperes áramok. Az egyik lehetőség az volt, hogy a szivattyú elektromos vezetékére egy másik vezetéket tekercselek. Finomhangolás után a tekercsben indukált áramot a processzor érzékelni tudta volna. Sajnos ezt a rendszert megépíteni – figyelembe véve az otthon rendelkezésemre álló eszközöket – túl sokáig tartott volna.
Jó néhány ötlet elvetése után a Google-höz folyamodtam segítségért. Kutakodásom közben véletlenül felidéződött bennem a Hall-hatásnak nevezett jelenség. A Hall-hatás a mágneses mezőben áramló elektronokra ható Lorentz-erő megnyilvánulása. A Lorentz-erő az elektromos és a mágneses mezőre egyaránt merőleges, hatására és irányában az elektronok eloszlása egyenetlenné válik. A vezető felületén indukált ez irányú feszültség arányos a mágneses mező erősségével, így alkalmas az erősségének a mérésére. A Hall-hatást rengeteg különféle, a kereskedelemben könnyen beszerezhető műszerrel mérni lehet, ezek a belső jelfeldolgozás milyenségében és a mágneses erők érzékelésének finomságában térnek el egymástól. Saját céljaimra az Allegro Microsystems A3240LUA típusa tűnt megfelelőnek. Ez egy kellően érzékeny, egypólusú érzékelő, adatlapját a ➲ http://www.allegromicro.com/sf/3240 címen meg lehet tekinteni. Az egypólusú érzékelők alapjában véve olyan NPN tranzisztorként viselkednek, amelynek akkor van bázisárama, ha az eszköz közelében déli mágneses pólus van jelen. Próbálgatás céljából beszereztem néhány ilyen érzékelőt.
Az volt az elképzelésem, hogy a távoli érzékelő mindössze a Hall-hatásra épülő eszközből állna, és ez a jelfeldolgozó általános be- és kiviteli lábára (GPIO) csatlakozna. A programoldal hibáinak a keresését különálló áramkörrel akartam megkönnyíteni, amely a szivattyú működését egy LED-del jelezte volna. Így legalább arról meg tudtam volna bizonyosodni, hogy az érzékelő valóban érzékeli-e az átfolyó áramot. Így készítettem el az *ábrán* látható áramkört.
Csatlakoztattam az elemeket, megmozgattam egy mágnest az érzékelő előtt, és a LED várakozásaimnak megfelelően kigyulladt. A következő lépés a szivattyú elindulásának a megvárása, majd az érzékelőnek a tápkábel mellett való mozgatása volt, ezzel megbizonyosodtam arról, hogy valóban képes a közelében lévő mező érzékelésére. Vártam, mozgattam, semmi… Vártam, mozgattam, még mindig semmi. Úgy tűnt, hogy a mező belső és külső határai közelebb voltak a vezetékhez, mint hittem. A fázis- és a nullavezetékek mágneses mezői elég erősek voltak ahhoz, hogy kioltsák egymást, így képtelen voltam mérni őket. Teljesen mindegy volt, hogy hova helyeztem az érzékelőt, az áramot nem lehetett mérni. Természetesen nem adtam fel ilyen könnyen, inkább módosítottam a terveimen. Egy régebbi, 15 amperre méretezett hosszabbítóval és egy lágyvas maggal felszerelkezve nekiláttam a mező felerősítésének: tízszer körbetekertem a nullavezetékkel a vasmagot. Művemet néhány kábelkötegelővel és némi ragasztóval tettem teljessé. Módosított kábelemmel ezt követően újra megindultam a pince felé. Csatlakoztattam a szivattyút a hosszabbítóhoz, majd vártam, amíg működésbe lépett. Miután a szivattyú újra bekapcsolt, végre sikerült az érzékelőt elég közel tenni a vasmaghoz, és érzékeltetni vele a mezőt. Kevéske falap, hangyányi forrasztás, még egy kis ragasztó, és máris készen állt a végleges érzékelő. A *képen* ez látható.
Következő lépésként el kellett döntenem, milyen processzort használjak a jelek feldolgozására. A fő szempont az ár, a beépített ethernetcsatoló, a használható GPIO-k és a Linux futtatásának lehetősége volt. Némi kutakodás után arra jutottam, hogy rengeteg olyan beágyazott mikrokontroller van, amely ethernetcsatolóval és a feladathoz elegendő teljesítménnyel egyaránt rendelkezik, ám a legtöbbnél nem említették kifejezetten a Linuxot. A kínálat másik végén a PC/104 osztályú beágyazott személyi számítógépek kellették magukat, igaz, jóval nagyobb teljesítménnyel és magasabb árral, mint amennyit én gondoltam el erre a célra. Végül a Soekris Engineering Net4501

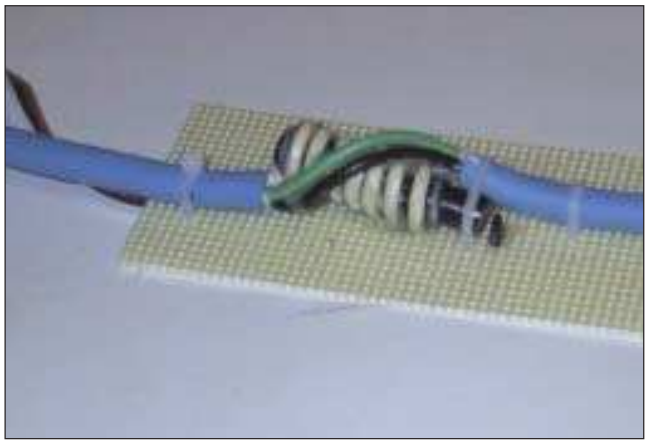


A próbaáramkör vázlata

jelzésű, egyetlen áramköri lapból álló számítógépe mellett döntöttem, amely CompactFlash foglalattal, 64 MB RAM-mal, AMD Elan processzorral és beépített ethernetcsatolóval rendelkezik. Érdekessége, hogy a Preboot Execution Environment (PXE) megoldás segítségével hálózatról is képes indítani az operációs rendszert.

A Soekris webhelyén (➲ http://www.soekris.com) elérhető leírásból megtudtam, hogy az Elan GPIO lábainak jelentős része egy csatlakozó révén – egy +5 voltos tápvonal társaságában – könnyedén hozzáférhető. Az ára elfogadható volt, és mellékeltek hozzá egy tápegységet, valamint egy pofás fémdobozt, amibe be lehet szerelni az áramkört. Az Elan GPIO lábai belső behúzással vagy elengedéssel működnek. Egy belső behúzással rendelkezőt választottam magamnak, mivel így az érzékelőt további alkatrészek beépítése nélkül, önmagában is rákötöttem a processzorra.

Ezután készítettem egy hálózati rendszerindításra képes (2.4.19-es) rendszermagot, amelyen szinte mindent letiltottam. A magot modulok nélkül fordítottam le, mindössze a Natsemi Ethernet illesztőprogramot, a gyökér NFS-t, a soros konzolt és az SC520 figyelő időzítőt hagytam meg. A normál beállítások megadása mellett egy további módosítást is végre kellett hajtanom a rendszermagon. A 2.4-es sorozatba tartozó rendszermagoknál x86 alapú gépekhez az alapértelmezett időzítőmegszakítás 100 Hz-re van állítva. Mivel tudtam, hogy egy közel ekkora frekvenciájú (60 Hz-es) jelet kell majd mintavételeznem, úgy döntöttem, hogy növelem az időzítőfrekvenciát. A megszakító időzítőfrekvenciáját az ***asm/param.h*** fájlon belüli Hz-érték szabályozza. Ennek felső határa 2000; én az 1500-as érték mellett döntöttem, vagyis másodpercenként 1500 megszakítást akartam kapni. Mivel a gépen más nem nagyon futott, nem volt komoly esély arra, hogy a megnövelt frekvencia miatt a megszakításokra alapuló eljárások összevesznének egymással. Az így létrehozott rendszermag elérhetővé tételének feladatát DHCP-kiszolgálómra és a PXELinuxra bíztam. Ekkor már csak a gyökérfájlrendszernek a TFTP-kiszolgálóra való átmásolása volt hátra. A kezdeti futtatókörnyezet létrehozásához lefordítottam a legújabb uClibc, BusyBox, TinyLogin és utelnetd csomagokat. Mindhárom futtatható állományt állandó jelleggel csatoltam az uClibc-hez. A BusyBox-féle init alapállapotban egy héjat indít a konzolkapun. A további szolgáltatásokat saját ***/etc/inittab*** állományom segítségével adtam hozzá a rendszerhez. Ez engedélyezi a konzolhéjat, meghív egy egyszerű parancsfájlt, amely (újra)csatlakoztatja a gyökérfájlrendszert, engedélyezi az időzítőt, majd a `telnetd` indításával lehetővé teszi, hogy távolról is beléphessek a gépre. Tehát egy terminált csatlakoztattam a soros konzolkapura, majd újraindítottam az eszközt. A konzolkapun keresztül figyelemmel tudtam követni a rendszer betöltésének a folyamatát, majd megjelent előttem a BusyBox-féle héj.



A kész érzékelő

Miután a rendszer életre kelt, figyelmemet az új illesztőprogramok felé irányítottam. Esetemben rendszermagterületen futó eszközillesztőre egyetlenegyre volt szükség, mégpedig annak a GPIO lábnak a figyeléséhez, amelyhez az érzékelőt csatlakoztattam. Mivel a rendszermag programozásában még nem voltam jártas, úgy döntöttem, hogy a lehető legkisebbre próbálom meg szorítani annak a valószínűségét, hogy a rendszermagban valami hiba lépjen fel, és ezért a lehető legkevesebb kódot írom meg. Ennek szellemében tehát egy ***/proc*** fájlrendszerbeli illesztőprogram írása mellett határoztam, amelynek feladatául a szivattyú be- és kikapcsolt állapotának a jelzését szántam. Ha ez az alacsonyabb szinten futó illesztőprogram elkészült, akkor egy felhasználói területen futó programmal már könnyedén lekérdezhető.

Az illesztőprogram init függvénye három lényeges műveletet hajt végre. Először a `create_proc_entry` hívással bejegyzi magát ***/proc*** fájlrendszerbeli modulként. A `proc_dir_entry` visszatérési értékeként két fontos adatszerkezetet kapunk, ezek a fájl- és fájlleíró műveleti táblák. Értékül két állandó, a megfelelő értékekkel feltöltött adatszerkezetet kapnak. Mivel meglehetősen egyszerű modulról van szó, a két adatszerkezet bejegyzései leginkább NULL értékeket tartalmaznak. A ***proc*** bejegyzés létrejötte után az `init` eljárás néhány, az Elan processzorra egyedileg jellemző regiszter értékének megadásával a kívánt GPIO-t állítja be bemenetként.

Az indítóeljárás utolsó lépésként egy időzítőt indít el, amely biztosítja a bemeneti láb rendszeres lekérdezését. Az időzítőfüggvényt érdemes jobban is megvizsgálni. (A listát lásd az 54. CD Magazin/Szennyviz könyvtárában.)

Mivel a mágneses mező – illetve emiatt az érzékelő jele is – oszcillál, nem lehetett egyszerűen mintavételezni a lábon bejövő jelet, és ezt a szivattyú állapotának jelzéseként kezelni.

Mivel el akartam kerülni, hogy a rossz időzítésű mintavételezés miatt zajok jelenjenek meg a statisztikákban, egy egyszerű integrátort valósítottam meg. Amikor az időzítő elindul, a mintaszámláló a periódusonként vételezni kívánt minták számával egyenlő értéket vesz fel. Alapesetben ennek értéke Hz/60, vagyis másodpercenként 25 minta. Ne feledjük, a Hz a rendszermag időzítő megszakításának a frekvenciája, amelyet a rendszermag lefordításakor 1500-ra állítottam.
A 25 gyorsminta beolvasása után újra beállítom az időzítőt, hogy a mintacsoport vételezési időtartamának a végén ismét lejárjon. Alapesetben öt másodpercenként olvasok be egy-egy mintacsoportot. Az időzítő újra történő beállításakor integrálom is (összeadom) a gyorsminták számát. Mivel a mintavételezés elég gyorsan történik, biztosan meg tudom határozni, hogy a szivattyú üzemel-e. Ha úgy tűnik, hogy a szivattyú be van kapcsolva, megváltoztatom a `pump_state` változó értékét. A modul adatainak olvasásakor (ezt a `pump_output` függvény végzi) mindössze ennek a változónak az állapotát vizsgálom meg és adom tovább. Ez a módszer lehetővé teszi, hogy úgy játszadozzak a mintavételezéssel, hogy közben ne kelljen az illesztőprogram válaszidejének a megváltozásától tartanom.
Az illesztőprogram hibáinak keresésekor egy `verbose` kapcsolót is beépítettem, amely különféle adatok kiírását teszi lehetővé a naplófájlba. A modult a `verbose=1` kapcsolóval futtatva például a mintavételi átmeneti tár tartalma íratható ki, amikor a szivattyú működni látszik. Így egy egyszerű oszcilloszkópfüggvénnyel a naplófájl tartalma alapján rendszeresen ellenőrizni tudom, hogy nem kapok-e hibás eredményt. Egy másik, ugyancsak érdekes lehetőség is kínálkozik. Mivel ismerem az érzékelő kioldási pontját és a pontok közötti fázisszöget (időt), ki tudom számolni a mágneses mező erősségének csúcsértékét az érzékelőnél. 1500 Hz-es órajelnél az érzékelő öt mintavétel idejéig van bekapcsolva. A mező 60 Hz-es frekvenciával oszcillál, vagyis a minták között 0,4 PI radián fázisszög van.
Az érzékelők kioldási pontjaira a legrosszabb esetet feltételezve – 50 Gauss felső és 5 Gauss alsó határérték – és az alábbi egyenleteket megoldva az amplitúdó csúcsértékére 51,4 Gausst kapok. Általános értékeket véve (35 és 25) a mező csúcsértéke nagyjából 38 Gauss:

```
A * sin( theta ) = 35
A * sin( theta + 0,4PI ) = 25
```

Ugyancsak a hibakeresést segítette egy másik illesztőprogram, amely egy másik GPIO-hoz csatlakoztatott LED-et kapcsol be, ha erre utasítom. Ez az illesztőprogram hasonló a szivattyúéhoz, ám ennek kimeneti függvénye – amely a modulról való olvasást teszi lehetővé – nem tesz szükségessé kifinomult mintavételezést, valamint bemenettel is rendelkezik, amely lehetővé teszi, hogy a felhasználó a modulra írva beállítsa a GPIO láb állapotát. Ez az új függvény (`led_input`) egy felhasználói területen található átmeneti tárból olvas, majd ennek alapján dönti el, hogy mire kell állítania a láb állapotát. A függvény bejegyzése a `file_operations` adatszerkezettel történik.
Ez az illesztőprogram a szivattyúhoz készítettől szerkezetileg annyiban tér még el, hogy a fájlra vonatkozó engedélyek között (ezeket a `create_proc_entry` hívásnál adjuk meg) lehetővé kell tenni az írási hozzáférést. Az illesztőprogram egy egyszerű parancsfájllal társítva alkalmas arra, hogy a szivattyúnál tájékoztatást adjon a programok működéséről: ha a LED be- és kikapcsolása követi a szivattyúét, akkor minden rendben van és működik.
Miután az alapvető illesztőprogramok a helyükre kerültek, a többi építőelem beillesztésével összeállhatott a rendszer. Szükség volt egy felhasználói térben futó démonra, amelynek segítségével távolról is le lehetett kérdezni a szivattyú állapotát. Mivel a gyökérfájlrendszer NFS-en keresztül van a központi kiszolgálómról befűzve, elég lett volna egy parancsfájlt készítenem, amely egy időre mindig elaludt volna, majd időnként ellenőrizte volna a */proc/pump* állapotát, és egy valódi fájlba írta volna a kapott eredményt. Ezúttal azonban a könnyebb megoldás helyett a `pumpserv` megírása mellett döntöttem.
A `pumpserv` egy egyszerű démon, amely az 5678-as kapun fogadja a kapcsolatokat, majd a */proc/pump* teljes tartalmát a másik fél felé másolja át. A csővezeték másik végén a `pumpwatch` helyezkedik el. A `pumpwatch` egy másik démon, amely a gazdagépen fut, és rendszeresen ellenőrzi a szivattyú állapotát, valamint rögzíti az állapotváltozások időpontját. Az átmenetekhez időbélyeget csatol, majd az adataikat egy naplófájlba írja. A naplófájlt később további programokkal fel lehet dolgozni, tartalma alapján kimutatásokat lehet készíteni, illetve valamilyen webhelyre feltöltve az egész világon elérhetővé tehető.
A rendszer 2003 áprilisa óta folyamatosan működik. Mivel a jelek szerint hibátlanul üzemel, nyugodtan sikerként könyvelhetném el, és túlléphetnék ezen a gondon, ám nem tudok szabadulni a `pumpserv2` megírásának a gondolatától. Ha valaha is nekiállok, egy pár dolgot másként fogok csinálni. A jelenlegi rendszer egyik súlyos hibája, hogy a gyökérfájlrendszert egy NFS-kiszolgálóról veszi, az adatokat pedig egy a kiszolgálón futó démon veszi át. Üzembiztosabb megoldást kaptam volna, ha a gyökérfájlrendszer helyi lenne, és mivel a Soekris Net4501 rendelkezik CompactFlash foglalattal, ennek megoldása nem is volna lehetetlen. Jó lenne, ha az adatok naplózását a szivattyú oldalán helyben végezné egy démon, majd kérésre elérhetővé tenné őket. Így a központi kiszolgáló leállása nem okozna adatvesztést.
Ugyanezzel a rendszerrel – apróbb módosítások árán – más eszközöket is lehetne figyelni, feltéve, hogy az érzékelő számára észlelhető nagyságú áramot vesznek fel. Néhány példa a sok közül: légkondicionálók, hűtőgépek, izzócsöves fűtőgépek, búvárszivattyúk. A leghasznosabb – és legfontosabb – alkalmazási lehetőség talán az irodai kávéautomatában található kávé mennyiségének a követése. A fűtőszálak bekapcsolásának száma ugyanis fordítottan arányos a még lefőzhető kávé mennyiségével.
Ha valaki a nyomdokomban haladva hasonló rendszert szeretne építeni, akkor az indító parancsfájlt és az illesztőprogramok, a `pumpwatch` és a `pumpserv` forráskódját a ➲ http://pumps.oldtools.org/src címről érheti el. A szivattyúfigyelő fájlrendszere ***.tar*** állományba tömörítve szintén hozzáférhető. Ha valaki tudni szeretné, hogy a pincémet fenyegeti-e árvíz, szivattyúm állapotát a ➲ http://pumps.oldtools.org címen ellenőrizheti.
A forráskód a Linux Journal FTP-helyéről, az ➲ http://ftp.ssc.com/pub/lj/listings/issue113/6827.tgz címről ugyancsak letölthető.

Linux Journal 2003. szeptember, 113. szám

Tad Truex
Napközben Alpha processzorok tervezésén dolgozik a HP-nál. Éjjel két gyermeke és számos hobbija között próbálja megosztani az idejét.

Szórakozás szünetmentesen...

APC szünetmentes tápegységek: széles körű használat, kimagasló linuxos támogatás.

Röviden bemutatom, milyen egyszerű üzembe helyezni egy USB csatlakozóval ellátott változatot (APC Back-Ups CS 500VA) Debian GNU/Linux alatt. Ha van soros kapu a szünetmentes tápegységen, akkor elegendő csupán az `apcupsd` csomagot feltelepíteni, és egynéhány alapvető beállítás után már használhatjuk is. A beállításokra a cikk végén térek ki részletesen.

Az `apcupsd` program 3.9.2-esnél frissebb változatai rendelkeznek USB-támogatással, ezek viszont nem érthetők el csomag formájában, így magunknak kell őket fordítanunk. A rendszermagnak támogatnia kell a következő eszközöket: `usb-hub` (`ehci`, `uchi`, `ohci` vagy amilyen van), `hid` (Human Interface Devices), `hiddev`, `usbdevfs`. A rendszermag beállításaiban az ***USB support*** menüpont alatt találhatók. A modulok lefordítása után a következő parancsot adjuk ki:

```
modprobe usb-uchi hid
```

Érdemes a */etc/modules* fájlba is beírni, hogy a rendszer elindulása során önműködően betöltődjenek. Ha mindezzel megvagyunk, a következő sorral kell kiegészítenünk a */etc/fstab* állományt:

```
none     /proc/bus/usb    usbdevfs
➥defaults,auto    0        0
```

A befűzéshez adjuk a ki a `mount -a` parancsot. Amennyiben a szünetmentes tápegység eszközfájlja még nem létezik, a következő parancsokkal létre kell hoznunk:

```
mkdir -p /dev/usb/hid
mknod /dev/usb/hid/hiddev0 c 180 96
```

Még ennél is egyszerűbb, ha az `acpsupsd` forrásában található *examples/make-hiddev* programmal hozzuk létre, ami eszközt is életre hív, mint amennyire szükségünk lehet. Másoljuk a */usr/src/linux* könyvtárba a pillanatnyi rendszermag forrását vagy a *<rendszermag_forras>/include/linux/hiddev.h* állományt az elérési úttal együtt – természetesen megfelel egy közvetett hivatkozás (symbolic link) is.

Telepítés

A ➲ http://www.apcupsd.com/ címről szereztem be a 3.10.6-os változatot. Csomagoljuk ki egy tetszőleges helyre, majd fűzzük be a szünetmentes tápegységet. Hogy megbizonyosodjunk róla, hogy vajon minden rendben lezajlott-e, indítsuk el a következő programokat a *<apcupsd_forras>/examples/* könyvtárban:

```
make hid-ups
./hid-ups
```

Ha hiba nélkül lefutott, az 1. listán (54. CD Magazin/Szunetmentes könyvtára) látható kimenethez hasonló eredményt kell látnunk. A CRTL+C-vel való leállítás után következhet a fordítás és a telepítés.

2. lista

```
doshutdown)
        echo "UPS ${2} initiated Shutdown
          ➥Sequence" | wall
        ${SHUTDOWN} -h now "apcupsd UPS
        ➥${2} initiated shutdown"
```

3. lista

```
doshutdown)
        echo "UPS ${2} initiated Shutdown
Sequence" | wall
      (who;w;df;dmesg;ps aux;netstat)|mail
➥-s "`hostname` gép szünetmentes tápegysége
➥lassan lemerül, leállítás következik."
➥kpeter@sysconfig.hu
      sleep 5
        ${SHUTDOWN} -h now "apcupsd UPS
          ➥${2} initiated shutdown"
;;
```

4. lista

```
~# /sbin/apcupsd
FATAL ERROR in linux-usb.c at line 615
Cannot open UPS device
```

```
./configure --enable-usb
make
make istall && echo Hurrá mindjárt kész vagyunk!\
```

A `configure` parancsnak a fent láthatón kívül több különböző kapcsolót is megadhatunk, például megmondhatjuk, hogy a telepített fájlok hova kerüljenek:

```
--bindir=DIR,
--sbindir=DIR ...
```

Ha másodlagos (slave) kiszolgálókat is használni akarunk, az alábbi kapcsolót írjuk be:

```
--enable-net
```

A szünetmentes tápegység és a kábel típusát, valamint az eszköznevet már itt is megadhatjuk (de később a beállítás-fájlban bármikor megváltoztathatjuk):

```
--with-upstype=TYPE
--with-upscable=CABLE
--with-serial-dev=DEV
```

Beállítások

A helyes működés érdekében két fájlt kell testreszabnunk: az *apccontro*l-t és az *apcupsd.conf*-t. Mindkettő alapértelmezés szerint a */etc/apcupsd* könyvtárban található.
A szünetmentes tápegységgel kapcsolatos beállításokat az *apcupsd.con*f tartalmazza. A CD-mellékleten megtalálható az eredeti és egy beállított változata is. Most csak néhány lényeges kapcsolót említek meg:

- *BATTERYLEVEL <töltés_%-ban>, MINUTES <minute>*: ha az akkumulátor töltése az adott határ alá esik, vagy az adott percnél kevesebb számított idő van hátra, akkor leállítja a számítógépet. Elegendő az egyiknek teljesülnie.
- *WAKEUP <másodperc>*: ha visszatér az áram, az adott ideig csak az akkumulátort tölti, és nem ad áramot a számítógépnek. Mindenképpen nullánál nagyobb értéket érdemes megadni, mert lehet, hogy pár másodpercre jön csak vissza az áram, és akkor ki-bekapcsolgatná a gépet.
- *SELFTEST <óra>*: megadott időközönként kipróbálja magát, hogy jól működik-e, tehát párszor átkapcsol akkumulátorra és vissza. Az alapértelmezett időtartam két hét (336).

Az *apccontrol* parancsfájl a démon által kiváltott események hatására meghívódik, és az eseménytől függően programokat indít el. Alapbeállításként csak körüzeneteket küld (broadcast message), illetve ha az akkumulátor energiája lecsökkent arra szintre, amit előre megadtunk, a `shotdown` programmal leállítja a számítógépet. Ez egy nagyon jól átlátható fájl, amit könnyen, a saját igényeinknek megfelelően át tudunk alakítani. Egy jellemző részlete a *2. listán* látható, ami a *3. listán* láthatóakhoz hasonlóan egészíthető ki.
Miután testreszabtuk, adjuk ki a következő parancsot, hogy érvényessé váljanak a beállítások:

```
/etc/init.d/apcupsd restart
```

Amennyiben a *4. listához* hasonló eredményt kapunk, akkor nincs jól beállítva az USB-támogatás. Érdemes kipróbálni, hogy minden úgy működik-e, ahogy gondoltuk.

A cikkhez tartozó listák megtalálhatóak az 54. CD Magazin/Szunetmentes könyvtárában.

Kolcza Péter (kpeter@sysconfig.hu)
Imádja a South Parkot. A Miskolci Egyetem informatika szakos hallgatója. Elvakult Linux-rajongó. Ha egyetemi elfoglaltságai engedik, Linuxszal és rendszerépítéssel foglalkozik.

KAPCSOLÓDÓ CÍMEK

- http://www.apcupsd.com/
- http://homepage1.nifty.com/Que/plamo/apc-ups/manual/usb.html