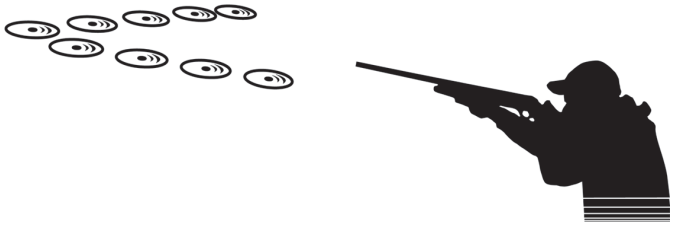
Beköszöntő

Csontos Gyula
a Linuxvilág szakmai szerkesztője.
szakmai témájú leveleiket örömmel
várja a következő levélcímen:
Csontos.Gyula@linuxvilag.hu

A linuxos társadalom ebben az évben is roppant nagy lépéseket tesz a kamaszkorból a felnőttkor irányába. A kernel.org oldalra kikerült a 2.6.0-s rendszermag test1-es változata, ami már elérhető közelségbe hozta a legújabb megbízható rendszermag-sorozatot. A 2.4-es sorozat megjelenésekor mindenki hatalmas várakozással tekintett a közeli jövőbe, lelki szemeink előtt nagyvállalati kiszolgálókon futó Linuxok lebegtek. Ez többé-kevésbé meg is valósult, a Linux hatalmas népszerűségre tett szert: mind a vállalati piacokat, mind az otthoni felhasználók szívét hódítgatja. A Microsoft most már ellenfelének tekinti a Linuxot és igyekszik mindent megtenni piaci versenyelőnyének megtartásáért. Eljött ez az idő is, sokan sokféleképpen gondoltuk ezeket a dolgokat, mind időben, mind pedig kiterjedésben. Szerencsére a Linux hihetetlen gyors fejlődésének köszönhetően, sokkal hamarabb bekövetkeznek hatalmas változások az elfogadása terén. Immár mi is rendelkezünk saját „nemzeti” Linuxszal, az UHU-Linux Kft. jóvoltából. Ezzel elhárult az ország meghódításának akadálya…

Programvadászat



Régen adtunk már közre teljes Linux-terjesztést, ezért a mostani mellékletünkön megtalálható a SoL – Server optimized Linux legfrissebb kiadása. Ezt a Linuxot az antitachyon cég fejleszti Bécsben. Mint neve is mutatja, ez egy kiszolgálóra készített Linux. Nagyon sok kiszolgálóprogram található a kiadásban, jelenleg a 16.00-s változatnál tart. Kifejezetten i686-os gépekre fordították, így a régebbi gépeken (például: i486, Pentium) ne is próbálkozzunk meg a telepítésével.

Az alábbiakban kedves olvasóinkat végigvezetem a SoL Linux telepítési menetén, ami sok tekintetben eltér a mostanság megszokott telepítéstől. Szinte csak a fájlok másolásából áll, beállítani mindent a későbbiekben kell.

A kezdetek

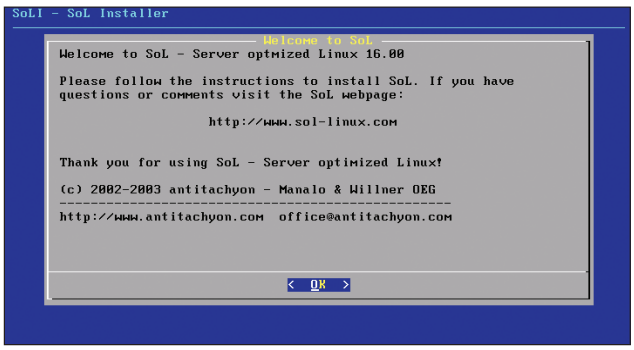
Helyezzük be a CD-t a meghajtóba, és indítsuk újra a gépet, ha a BIOS-ban be van állítva, hogy a rendszer indítása CD-ROM-ról történjen, akkor a telepítő azonnal elindul. Különféle indítási módok közül választhatunk, vagy egyszerűen csak üssünk ENTER-t, és kezdjünk neki a telepítésnek.

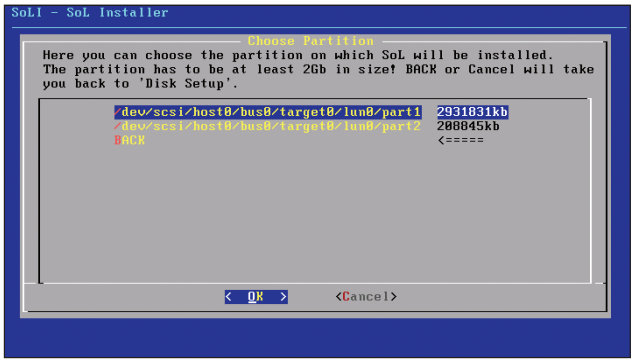
`install` – egyszerű telepítés.
`installscsi` – SCSI-meghajtókkal kiegészített telepítés.
`diag` – különálló rendszerindításra alkalmas lemezkép.
`diagscsi` – különálló rendszerindításra alkalmas lemezkép, SCSI-meghajtókkal.
`memtest` – memóriatesztelő program indítása.

Indítsuk el a telepítést!

Első lépésként a merevlemez kiosztását kell elvégeznünk, ehhez a már jól megszokott `cfdisk` programot adja a telepítő. Készítsünk egy legalább 2 GB-os lemezrészt a SoL Linuxnak, ugyanis ennél kisebb helyen nem érzi jól magát. Miután ezzel végeztünk, válasszuk ki a megfelelő lemezrészt, és formázzuk meg. Ezen a ponton `ext`, `ext3` vagy a ReiserFS közül lehet választani, tapasztalataim szerint az `ext3`-mal a telepítő nem képes hiba nélkül végigfutni.

A telepítés ezután kezdődik, ami igazából fájlok sokaságának végeláthatatlan másolásából áll. Valójában a telepítő 15 percig ír (majdnem ennyi idő alatt végzett is), de nekem egy örökkévalóságnak tűnt.





Ezután egy gyors hálózati beállítás következik: a gép nevét, a tartománynevét, az IP-címét és a névkiszolgáló nevét kell megadni neki.

A LILO telepítése után a rendszer újraindítása következik.

Újraindítás után

Mivel a telepítés alatt nem adtunk meg rendszergazdai jelszót, első feladatunk még a rendszerindítás alatt ennek pótlása lesz. A rendszer telepítésével végeztünk is, most jön a nehezebb feladat, a beállítás. Én személy szerint nagyon örültem ennek a „fapados” Linuxnak. Olyan érzésem volt, mintha visszapottyantam volna a régi szép időkbe, a jóféle rendszertelepítésekhez: nincs grafikus eszköz, amiről nem tudja az ember, hogy mit is csinál, hanem ha valamelyik beállítófájlba beleírok valamit, akkor az úgy fog működni, ahogyan én szeretném.

Beállítások

A rendszer beállításai XML-alapokon nyugszanak, a legtöbb beállításhoz tartozó fájlt a */etc/rc.d/conf* könyvtárban találhatjuk. A kiszolgálóprogramok pedig a */server* könyvtárban kaptak helyet, találhatunk itt Apache, Apache2, OpenLDAP, PostgreSQL, MySQL, Samba, Squid, DHCP-kiszolgálókat – mindegyik a saját könyvtárában lévő beállítófájlokban szabható testre.

Rendszermag 2.6.0-test1

Végre elérhető közelségbe került a legújabb rendszermag sorozat, ezzel már bátran ismerkedhetünk. Sok újdonság található benne, érdemes elkezdeni a beállítások próbálgatását. A tapasztalataim azt mutatják, hogy minden gond nélkül lefordítható és a rendszer elindítható vele. A modulokat sajnos nem tudtam használni, ezért minden meghajtót a magba kellett fordítanom, így kiválóan működött minden egység, legyen az asztali PC, vagy a SiS630-as lapkakészlettel megáldott notebookom. Honlapjuk a ➲ http://www.sol-linux.com címen érhető el.

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

A hátsó ajtón osonva

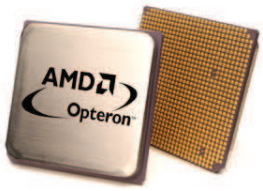
A Washington Post egyik írása szerint az amerikai cégek ugyan jelentős erőforrásokat fordítanak számítógépes rendszereik védelmére, egy, csak első ránézésre jelentéktelen tényező általában elkerüli a figyelmüket, ez pedig a vállalati telefonrendszer. Egyre több olyan esetet jelentenek, amikor a betörők a hangposta- vagy a telefonrendszert támadják meg, majd illetéktelen hozzáférésüket felhasználva más városokban vagy országokban üzemelő internetszolgáltatókhoz indítanak távolsági hívásokat. Így névtelenül tevékenykedhetnek, a szolgáltató pedig csak annyit észlel, hogy a telefonhívást a megtámadott cég rendszeréből indították. Mivel az újabb vállalati telefonhálózatok a számítógépes hálózatokkal is össze vannak kapcsolva, roppant vonzó célpontot kínálnak. A számítógépes rendszer védelme hiába megfelelő, ha a telefonhálózat felől behatolva könnyedén lehet adatokat lopni vagy károkat okozni úgy, hogy eközben a támadó teljes névtelenségbe burkolózhat.

Párosan szép

A Sun Microsystems olyan kettős megoldást mutatott be, ami a Linuxot és egy valós idejű mikrorendszermagot házasít össze. A Jaluna nevű, amúgy nyílt megoldás lényege az, hogy valós idejű jellemzőkkel ruházza fel a Linux alapú eszközöket úgy, hogy magát a Linux-rendszermagot nem kell módosítani. A cég megoldásának használatakor a fizikai processzor virtuális processzorként látszik a rajta futó rendszerek és programok számára, a memórián pedig lényegében két rendszermag osztozik. A valós idejű futást igénylő programokat – amolyan „programból megvalósított társprocesszorként” – a Sun C5-ös valós idejű rendszermagja kezeli, míg az általános célú alkalmazások Linuxon futnak. A Linuxot többen is próbálták, próbálják valós idejű képességekkel felruházni, ám ez egyre több ág kialakulását eredményezi, és a módosított részek miatt a fő vonal fejlődését is nehezebb követni – a Jaluna által éppen ezek a nehézségek hárulnak el. A Jaluna háromféle változatban mutatkozott be. A Jaluna-2/EL egy egyszerű beágyazott Linux-változat, a Jaluna-2/RT egy beágyazott Linux és egy szigorúan valós idejű működésre képes C5 mikrorendszermag párosítása, míg a Jaluna-2/VL esetében a C5 mikrorendszermag virtuális Linux-példányok vezérlését végzi, amelyek természetesen ebben az esetben is ugyanazon az eszközön futnak.

S eljött az Opteron

Számos határidő-csúsztatás után úgy néz ki, végre a felhasználókhoz is eljut az AMD 64 bites processzora, az Opteron. Az AMD elsősorban azt a piaci területet célozza meg új termékével, amelyen a jelenléte elhanyagolhatónak mondható: a kiszolgálók és a munkaállomások szegmensét. Az Opteron többféle változatban létezik, a 100-as sorozat az egyprocesszoros munkaállomásokba készül, a 200-as sorozat tagjait kétprocesszoros, míg a 800-as sorozat képviselői a négy- és nyolcprocesszoros gépekbe szánják.



Az AMD új lapkái viszonylag elfogadható áron kaphatók, például a 840-es modell 749 dollárba kerül, a 140-es modellért pedig mindössze 229 dollárt kell fizetni. Részletesebben minderről a 22. oldalon olvashatunk.

➲ http://www.amd.com

A G5 lett az első

Az Apple és az IBM közösen mutatta be a világ első 64 bites asztali processzorát, a PowerPC G5-öt. Még valamiben világelső, vagy legalábbis annak mondja magát az Apple, mégpedig abban, hogy az új processzor köré épülő Power Mac G5 gépe a világ leggyorsabb személyi számítógépe.

A G5 processzor természetesen már 64 bites, de a 32 bites alkalmazások natív futtatására is képes. Legerősebb példányának órajele 2 GHz, elvileg 18 hexabájtnyi memóriaterületet tud megcímezni. A G5 képes a párhuzamos feldolgozásra, két lebegőpontos egységgel rendelkezik, és teljes értékű szimmetrikus többprocesszoros támogatással bír. A lapka 130 nanométeres „szilícium a szigetelőn” eljárással készül, összesen 58 millió tranzisztort tartalmaz. A rendszervezérlővel 1 GHz órajelű összeköttetésen keresztül tartja a kapcsolatot, így másodpercenként akár 8 GB adatot is meg tud mozgatni. A DDR memóriával egy 128 bites, 400 MHz órajelű csatlakozáson keresztül kommunikál, HyperTransport felületének órajele 800 MHz, illetve az AGP 8x Pro grafikai csatoló felületet is támogatja.

Mondtam már, hogy betörtek hozzánk?

Az Egyesült Államokban új, figyelemre méltó folyamatokat érzékeltető törvény született. A Security Breach Information Act (A biztonság megsértéséről való tájékoztatásról szóló törvény) értelmében mindazok a vállalkozások, amelyek Kaliforniában folytatják tevékenységüket, vagy van kaliforniai az ügyfeleik között, kötelesek vevőiket tájékoztatni, ha számítógépes rendszerüket eredményes támadás érte, és ennek során az ügyfelek adatai is elérhetővé váltak. A törvénytől azt várják, hogy ügyfeleik adatainak gondosabb kezelésére bírja a cégeket, amelyek – ha az adatok biztonságának sérüléséről nem tájékoztatják az érintetteket – jogi eljárás elébe néznek. Valószínűleg az is fontos ösztönző erőt jelent a biztonság növelésében, hogy a betöréseket a továbbiakban – törvényesen – nem lehet majd eltitkolni, így a kellő gondosság nélkül eljáró vállalatoktól az adatvédelmi szempontból tudatos ügyfelek el fognak fordulni.

Egy próbát megér

A Lindows.com bejelentette LindowsCD nevű termékét. A LindowsCD a legújabb, 4.0-s számot viselő LindowsOS-t tartalmazza, és telepítés vagy különösebb beállítások végrehajtása nélkül teszi lehetővé a Linux kipróbálását.

A CD-ről futó terjesztés a legújabb Lindows-változat szinte minden szolgáltatását biztosítja, vagyis nemcsak felismeri az őt futtató gép összetevőit, illetve teljes körű, azonnal használható (plug and play) támogatást nyújt, de multimédiás alkalmazásokat és irodai csomagot is találunk benne.

A LindowsCD-t azoknak szánják, akik ki szeretnék próbálni a Linuxot, ám külön gépet nem tudnak ilyen célra biztosítani, illetve meglévő – jellemzően Windowst futtató – gépük merevlemezének a tartalmát sem szeretnék megbolygatni. Az ötlet nem új, nem is rossz, az viszont furcsa, hogy a felhasználók elhódítására szánt terjesztést normál körülmények közt harminc dolláros áron kínálják. Igaz, aki ezt nem akarja kifizetni, a pillanatnyi akció keretében a LindowsOS 4.0-hoz mellékelve most ajándékba is megkaphatja, illetve különféle kedvezményeket vehet igénybe – csakhogy a csábítás trükkjei között elvileg nem szerepel az ügyfél pénztárcájában való turkálás, annak később kell következnie.

➲ http://www.lindows.com/buyLCD

SQL-Legder

Az SQL-Ledger olyan integrált, ingyenesen használható ügyviteli rendszer, amivel a cégek könnyen és rugalmasan végezhetik el a megrendelések, a számlák feldolgozását, illetve azok egyidejű kettős könyvelését, valamint figyelemmel kísérhetik készletmozgásaikat. A program felhasználása rendkívül sokrétű, működik Windows, Linux, Macintosh operációs rendszer alatt, és a legfejlettebb adatbázismotorokat (Postgres, Oracle) használja. Önálló gépeken, helyi hálózaton, igény szerint Interneten keresztül is használható. Fontosabb jellemzői:

- ***Kiszolgálóalapú felépítés***
  Az adatok SQL adatbázisban tárolódnak (kiszolgáló), amit szabványos böngészővel (ügyfélprogram) lehet elérni. Nemcsak grafikus böngészőkkel lehet használni, hanem karakteres változatokkal is. A fejlett adatbázis-kezelésnek köszönhetően az adatokkal egyszerre több felhasználó dolgozhat, és tranzakció-kezelésének köszönhetően váratlan események (pl. áramszünet) sem befolyásolják a tökéletes adatmentést.
- ***Könnyen kezelhető szerkezet***
  A program nyílt forrású, tehát ingyenesen használható, a programkód megtekinthető és szabadon módosítható.
- ***Valós idejű könyvelés***
  A program valamennyi tranzakciójához előzetesen beállított főkönyvi számlák kapcsolódnak, így bármilyen gazdasági esemény rögzítésével a könyvelés is önműködően zajlik.
- ***Szerkeszthető sablonok***
  A program előre elkészíthető sablonokból hozza létre a számlákat, szállítóleveleket, megrendeléseket, pénzügyi kimutatásokat (mérleg, eredménykimutatás).
- ***Testreszabható kimutatások***
  Választható oszlopok, többféle rendezési lehetőség, részösszegek sorok megjelenítése, valamint többféle szűrésre nyílik lehetőség.

A program akárhány ügyfél, illetve cég ügyvitelének a vezetésére képes, és mindegyik ügyféladatbázishoz korlátlan számú felhasználó kapcsolható, különböző jogosultságokkal. A különböző ügyféladatbázisokhoz és felhasználókhoz saját sablonok, programfelületek, illetve nyelvi beállítások tartozhatnak. Az ***Audit kontroll*** szolgáltatás segítségével – a dátum beállítása révén – a zárolt adatok változatlansága bizotsítható.

➲ http://www.investor.hu/sql-ledger/login.pl?action=login&login=demo

Azonnal oldódó adatbázis

A Pogo Linux – linuxos eszközöket gyártó cég – és a MySQL AB bejelentette, hogy közösen fejlesztik ki az első MySQL alapú adatbázis-kezelő eszközt – egy olyan előre összeállított, megfelelő vassal és programokkal felszerelt készüléket, ami kifejezetten adatbázis-kezelést végez. A DataWare 2600 Serverből a két társaság együttműködése révén várhatóan a legjobb megbízhatóságot és teljesítményt sikerül majd kihozni, miközben telepítése gyors, használata egyszerű lesz. Mivel a gép nyílt forrású megoldásokra épül, ára jóval kedvezőbb lesz az ilyen jellegű, de zárt eszközökénél. A Pogo Linux hasonló termékeket más területekre is kínál, így StorageWare néven nagy méretű tárhelyet biztosító, Serial ATA alapú fájlkiszolgálókat, WebWare néven pedig előre összeállított webkiszolgálókat forgalmaz.

➲ http://www.pogolinux.com

Újabb Magnia-fivér

A Toshiba bemutatta a Magnia sorozat legújabb tagját, a Magnia SG30 Wireless Mobility Servert. Az elsősorban kisebb irodáknak szánt kiszolgáló a sorozat többi tagjához hasonlóan fájlkiszolgálóként, nyomtatómegosztásra és az internet-hozzáférés megosztására használható. Érdekessége, hogy beépített, IEEE 802.11b szabvány szerint hozzáférési ponttal rendelkezik, a bővítőkártyákat pedig három PCMCIA foglalattal fogadja. 1,2 GHz-es Intel Celeron processzorán Red Hat 8.0-s operációs rendszer fut, ami 256 MB ECC SDRAM-mal gazdálkodhat. A gép 128 bites, szabványos titkosításra képes a vezeték nélküli kapcsolatokon, továbbá támogatja a virtuális magánhálózatokat és a MAC-cím alapú szűrést. Ára 1500 dollár, vagyis hozzávetőlegesen 350 ezer forint.

Biometriai adatok az útlevélen

Az Európai Unió bizottsága olyan útlevél bevezetését tervezi, ami egy lapkában biometriai adatokat, így ujjlenyomatot és íriszmintát fog tárolni. A biometriai adatok révén az útlevelek eredetiségének ellenőrzése jóval könnyebbé válna, és az utóbbi években jelentős méretet öltött hamisításuk is nehezebb volna. Hasonló lapkák kerülnek majd az Európai Unión kívülről érkezettek vízumaira is, mozgásuk követését segítendő. Bár a megvalósítás pontos időpontját még nem tűzték ki, abban már sikerült megegyezni, hogy az EU teljes területén egységes rendszert kell bevezetni. Az EU 140 millió eurót különített el a biometriai kutatásokra és fejlesztésekre.

Gigabit minden asztalra

Az SMC új termékcsaládot jelentett be, amellyel a cég szándékai szerint minden munkaállomásra Gb/s sebességű hálózati kapcsolat kerülhet. A családba egyelőre két 10/100/1000 Mb/s sebességű hálózati kapcsoló tartozik, egy öt- és egy nyolckapus, illetve egy 32 bites, PCI-foglalatba illeszkedő hálózati kártya. A két kapcsoló 135 és 195 dolláros áron lesz kapható, tehát filléres dolognak semmiképpen nem nevezhetők, míg a hálózati kártyát barátságos 30 dollárért lehet majd megszerezni.

➲ http://www.smc.com

Majd írj a vonatról

Az Intel, a Bell Canada és a VIA Rail Canada együttműködésének köszönhetően a VIA szerelvényein Montreal és Toronto között utazók hamarosan vezeték nélküli internetkapcsolatot is használhatnak. Ez lesz az első alkalom, hogy Észak-Amerikában mozgó utaskocsit vezeték nélküli helyi hálózati kapcsolattal látnak el, és valószínűleg a világ többi részében sem találni túl sok példát ilyesmire. Az internetkapcsolatot – amit a felmérések szerint az utazó üzletemberek 90 százaléka igényel – műholdon keresztül biztosítják. A megfelelő felszereléssel rendelkező üzletemberek így már nemcsak a repülőtereken vagy a szállodákban férhetnek hozzá leveikhez vagy a vállalati magánhálózathoz, de vonatról is. Vagyis senki nem ússza meg, hogy az amúgy is fárasztó üzleti utakat aktív munkával töltse – a fejlődést nem lehet megállítani.

Egységes adatvédelmi szabályzat vállalatoknak

Az IBM olyan új programozási nyelvet mutatott be, amelynek segítségével a vállalatok az összes számítógépes alkalmazásra és rendszerre vonatkozóan önműködő módszerrel tartathatják be adatvédelmi házirendjüket. Az Enterprise Privacy Authorization Language XML alapú, a W3C P3P (Platform for Privacy Preferences) szabványára épül, ennek előírásait veszi át, és tartatja be az összes vállalati alkalmazással és adatbázison. Az IBM hamarosan szabványtervezetként is elő kívánja terjeszteni megoldását, és támogatása a cég saját adatvédelmi felügyeleti programjában, az IBM Tivoli Privacy Managerben is megjelenik majd. Nyílt forrású EPAL környezet Privacy Authoring Editor néven bárki számára hozzáférhető a

➲ http://www.sourceforge.net címről.

➲ http://www.ibm.com

3Com: minősítve

A 3Com bejelentette, hogy két terméke, a 3Com Firewall PC Card és a 3Com Embedded Firewall Policy Server a Common Criteria minősítési eljárás szerint EAL2+ minősítést nyert. A 3Com termékeit az amerikai hadsereg emberei is tesztelték, és nemcsak biztonsági szempontból találták megfelelőnek őket, hanem kellően felhasználóbarátnak és könnyen telepíthetőnek is bizonyultak. A nemzetközileg elfogadott minősítés tanúsítja, hogy szolgáltatásaikat és felépítésüket tekintve egyaránt alkalmasak arra, hogy védjék tulajdonosaik adatait. Az EAL1 minősítés hivatott igazolni, hogy az adott termék szolgáltatásai valóban a tervezők szándékai szerint működnek, míg az EAL2 minősítés megléte garanciát jelent arra nézve, hogy a termék tervezése során megpróbálták megkeresni az eszköz sebezhető pontjait, illetve a későbbiekben felismert hiányosságok javítására megfelelő eljárásokat dolgoztak ki.

➲ http://www.3com.com/security

Via Antaur

A Via Technologies bemutatta új, mobilgépekbe szánt processzorát, ami a mitológiai szereplőkre emlékeztető Antaur nevet kapta. A kis fogyasztású Antaur elsőként Kínában jelenik meg a vékony és könnyű hordozható gépekben, majd lassacskán Európába és Amerikába is eljut. Az Antaur a C3 „Nehemiah” magra épül, legnagyobb fogyasztása 11 Watt, ezt a PowerSaver 2.0 megoldás feladata további ötven százalékkal csökkenteni. Az új processzor képes a programszálakban lévő elágazások előzetes becslésére, 16 csővezeték-állomással rendelkezik, támogatja az SSE utasításokat és 64 kB másodszintű gyorsítótárral látták el. Található benne egy Via PadLock adattitkosító motor, illetve egy véletlenszám-generátor is, ezek a biztonsági alkalmazások futását segítik.

Elhozzák a csillagokat

A SuSE Linux egy 2,4 millió eurós tervezet részeként szerződést kötött a Német Űrkutatási Központtal. A tervezet keretein belül egy nagyteljesítményű tárolórendszert építenek a nemzetközi űrállomáshoz kapcsolódó Columbus, az európai űrlabor számára, amelynek fellövését a jövő évben tervezik. A CC CompuNet, Németország legnagyobb független informatikai szolgáltató cége és a SuSE Linux közösen biztosítja a Columbus programrendszerét.

Az űrlabor a tervek szerint az elkövetkező 10–15 évben a nap 24 órájában folyamatosan nagy mennyiségű telemetriai, hang- és videoadatot továbbít majd a Földre. A tárolásról, a fennakadásoktól mentes adatcseréről és az archiválásról többek között 46 db kétprocesszoros, Xeon alapú IBM gép, két Sun Solaris rendszer, tárolókiszolgálók és szalagkönyvtár gondoskodik majd, operációs rendszerként pedig SuSE Linux Enterprise Server 8-at alkalmaznak.

➲ http://www.esa.int

Áram a hálóról

Az IEEE 802.3af jelzéssel hivatalosan is szabványként fogadta el azt a szabálygyűjteményt, amely az elektromos táplellátás ethernetkábeleken való továbbításának a módját írja le. A szabvány véglegesítése a várakozások szerint számos új termék megjelenését teszi majd lehetővé, és hamarosan az számít különlegességnek, ha egy ethernetcsatlakozó nem lesz alkalmas elektromos energia biztosítására. Természetesen szó sincs arról, hogy a jövőben a betonkeverőket is ethernethálózatról fogjuk üzemeltetni, ám a kisebb fogyasztású eszközök, például hordozható számítógépek esetében mindössze a hálózati kapcsolatról kell majd gondoskodni, az áram „jön vele”. A kérdés már csak az, hogy mi lesz a másik húzóágazat termékeinek, a vezeték nélküli hálózatoknak a kedvelőivel.

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

Rendszermag-fejlesztési hírek

Mike a szabad programokat fejlesztő szakemberek példaképe volt, megkövetelte a programok kiválóságát és nyíltságát, és kész volt továbbfejleszteni, ha valami tökéletlen volt.

Új illesztőprogramok bukkantak fel a rendszermagban, amelyek kihasználják a korszerű, nagysebességű soros kapukat. A SuperIO lapkák már régóta támogatják a 460,8 Kbaud, illetve a 921,6 Kbaud sebességeket, de a Linuxhoz nem volt illesztőprogram. 2003 februárjában *David Woodhouse* küldött egy foltot, ami lehetővé teszi, hogy Linux alól is kiaknázzuk e lapkák képességeit.
Mostantól több ezer merevlemezt lehet használni Linux-környezetben. Korábban 256 volt a felső határ. *Badari Pulavarty* 2003 márciusában küldte be azt a foltot, ami jócskán felemeli ezt a lécet.
A próbafuttatás során sikeresen írt és olvasott négyezer különálló lemezt. A Linux által támogatott lemezek száma új felső határának megállapítása még folyamatban van. Több erőforrás is túlterhelődik a lemezek számának növekedésével, a legfontosabb ezek közül a memóriahasználat. A nagyszámú lemez használata mellett felmerülő memóriaszükséglet kérdése jelenleg is pezsgő kutatási terület. Végül is kiderülhet, hogy a lemezek számának létezik egy elméleti felső határa, de addig még sok szemetet el kell takarítani.
Új GPL-es grafikus rendszerindító jelent meg 2003 áprilisában, ami a Gujin nevet viseli, és a LILO felváltását célozza. *Etienne Lorrain* a nulláról indulva írta meg a programot. A Gujin a gép indításakor képes felismerni a betölthető rendszermagokat, ellentétben a LILO-val, aminél a betöltőszektort minden rendszermag-telepítéskor módosítani kell. Képes különlegesen nagy rendszermagok betöltésére, ami a 2.5-ös rendszermagsorozatnál fontossá válhat – ez ugyanis már feszegeti a LILO méretkorlátait. A betöltendő rendszermagot vonzó grafikus felületen választhatjuk ki.
David Brownell új API-függvénykészletet készített az USB-eszközök számára. Ezek elsősorban Linuxot futtató PDA-kban és más beágyazott rendszerekben nyerhetnek felhasználást. Az API-függvények egységes felülettel bírnak, ennek köszönhetően az egyik eszközre írt program a másikra is átvihető, annak ellenére, hogy a két vas esetleg teljesen eltérő.
Az új, *Lennert Buytenhek* írta program segítségével, az új *libivykis* programkönyvtár használatával mostantól hordozható módon nyomon követhető az állományok kivitelre, illetve bevitelre való készsége. Idén áprilisban jelentette meg a programkönyvtárat, ami egy burkoló a `poll()`, a `kqueue()` és más hasonló függvények köré, egységes felületet biztosít a nagyteljesítményű hálózati kiszolgálók számára, a rajtuk futtatott rendszertől függetlenül.
A *libivykis* SourceForge-oldala azt állítja, hogy a programkönyvtár annak ellenére teljesen megbízható és élesben használható, hogy az első nyilvános kiadás csak nemrég zajlott le. Akárhogy is, a *libivykis* programkönyvtárat már több protokoll proxykiszolgálója és videofolyamot szolgáltató kiszolgáló készítésére felhasználták.
A zárt kódú BitKeeper egyre inkább a rendszermagfejlesztés hivatalos változatkezelő rendszerévé válik. Az új valós idejű BitKeeperből CVS-be átalakító átjáró – amelyet *Larry McVoy*, a BK írója állított fel – sok BitKeepert ellenző rendszermagfejlesztőt elhallgattatott. Ezzel egy időben a szabad választási lehetőségek, például az arch és a Subversion összefogni látszanak, hogy közösen oldják meg azokat a súlyos gondokat, amelyek megakadályozzák, hogy a szabad változatkezelő rendszerek is azokat a világszínvonalú megoldásokat alkalmazzák, mint a BitKeeper.

Zack Brown

Linux Journal 2003. június, 111. szám

In memoriam

Március 28-án, pénteken, életének 38. évében elhunyt *Mike Jackson*, a Linux Shadow Password HOWTO szerzője.
Az elmúlt években Mike az LPSH karbantartásán kívül tengeralattjáró-felderítő készülékek használatát oktatta, tartalékos rendőrtiszt volt, valamint a TSCNet elnöke és alapító tagja, a Kitsap Peninsula Linux User Group alapító tagja és pénztárosa, továbbá részt vett a PHP SNMP modul és az OpenZaurushoz készített Sharp fényképezőgép-csomag fejlesztésében.

A Linux Shadow Password HOWTO utoljára 1998-ban frissült. Nem mintha Mike elvesztette volna az érdeklődését a téma iránt; éppen ellenkezőleg: a HOGYAN minden fejlesztőt meggyőzött, hogy terjesztéseikbe alapértelmezettként tegyék bele az árnyékjelszavak támogatását.
Ha Mike újonnan csatlakozott egy fejlesztéshez, rögtön fejest ugrott a munkába. Ha hibákat talált, kijavította őket, és visszaküldte a javításokat a karbantartóknak. Ha valamit hiányolt a programból, saját maga hozzáadta. Mike a szabad programokat fejlesztő szakemberek példaképe volt, megkövetelte a programok kiválóságát és nyíltságát, és kész volt továbbfejleszteni, ha valami tökéletlen volt.
Mike halála mélyen érintette családját és barátait, nagyon fog hiányozni mindenkinek. Összegyűjtöttük honlapjának a darabjait, és más hivatkozásokkal együtt elhelyeztük a ➲ http://www.kplug.org/~mhjack címen.

Brian C. Lane

Linux Journal 2003. június, 111. szám

Linux-index

1. A betegek és idősek ellátására tervezett Linux alapú japán robot, a Wakamuru magassága centiméterben: 100
2. A Wakamuru szókincsének nagysága: 10 000 szó
3. A meglévő rendszerek fenntartására költött IT-költségek százalékban: 79
4. A megtakarítás százalékban, ha Unix helyett Linuxot használnak az Oracle futtatására: 45–80
5. A nyílt forrású fejlesztésekre 2003-ban ennyi milliárd jent irányzott elő a japán Gazdasági, Kereskedelmi és Ipari Minisztérium: 1
6. A következő pénzügyi évre ennyi millió jent terveznek elkölteni azokra a kutatásokra, amelyek a japán kormányzat által használt számítógépek nyílt forrásra való átállásának lehetőségét vizsgálják: 50
7. Az SGI nagygép értéke amerikai dollárban, amelyet 12 Red Hat Linuxot futtató, telepbe szervezett PC-vel cseréltek le a Johnson Űrközpontban, Houstonban: 1 600 000
8. Az SGI nagygép éves karbantartási költsége amerikai dollárban: 50 000
9. A 12 PC és operációs rendszerük költsége amerikai dollárban: 25 000
10. Az első tíz webszolgáltató közül ennyi futtat Linuxot: 9
11. 2002-ben ennyi százalékkal csökkentek az IT-kiadások az IDC szerint: 4,1
12. 2003-ban a vállalatok ennyi százaléka tervezi növelni az IT-kiadásait az IDC szerint: 85
13. Ennyi százalékkal esik vissza az átlagos vállalkozások számítógépre és programokra fordított kiadása 2003-ban a Goldman Sachs szerint: 1
14. Előreláthatóan ennyi százalékkal növekszenek az IT-kiadások 2003-ban a Forrester Research szerint: 1,9

Források

1–2.: Linux Devices
3–4.: Oracle
5–6.: Associated Press
7–9.: Information Week
10.: Netcraft
11–12.: International Data Corp.
13.: Golman Sachs
14.: Forrester Research

Linux Journal 2003. június, 111. szám

Ők mondták

Az egyetemek nagyon költségérzékenyek, és az Intel alapú rendszerek viszonylagos alacsony költsége a Linux sokoldalú felhasználhatóságával összekötve hamar szabvánnyá válik, különösen a jó hírű mérnökiskolákban. A forráskód szabad elérhetősége arra sarkallja az egyetemeket, hogy számítástechnikai tantárgyaikat Linuxra alapozzák. *(Billy Marshall, Red Hat)*

A 3-as változat alapjainak fejlesztését főleg azok a vállalatok pénzelik, amelyeknek *Hans Reiser* felhasználói engedélyeket adott el a GPL mellett. Ezek a vállalatok nem akarják tudni, hogy zárt kódú termékük a ReiserFS-en alapul. Az ügyvédem azt kérdezte: „Az emberek fizetnek neked ezért?" Aha. Hi-hi. Az élet szép. Ha megveszed a ReiserFS-t, a saját feladatodra összpontosíthatsz, ahelyett, hogy az egész állományrendszert újra feltalálnád. *(mkreiserfs, a reiserfsprogs 3.6.5-ban)*

Az első felfedezés, amit most be szeretnék mutatni, egy lustáknak való algoritmus tudományos cikkek előkészítésére. Írd le, amit akarsz, és ne idézz senkitől semmit. A sértődött olvasók az összes olyan cikk adatát el fogják küldeni, amikre hivatkoznod kellett volna. *(Paul Graham)*

A Unix és Linux összehasonlítása azt az eredményt hozta, hogy egy Intel alapú számítógép 2–5-ször nagyobb teljesítményt (üzenet/másodperc) nyújtott, mint a Sun Sparc számítógép, fele árért. *(Casey Merkey – Global Linux Program Manager, Reuters' Market Data System)*

Sajnálatos módon 2003 nem a vállalati, linuxos asztali gépek éve lesz, de a nagy rendszergyártók (a Dell, a HP, az IBM és a Sun) várhatóan egyre inkább az asztali gépeken futó Linuxot támogatják (amire 2002-ben gyakorlatilag ügyet sem vetettek), mivel felismerték, hogy több Linux-kiszolgálót tudnak eladni, ha létezik használható asztali gépes Linux. *(Aberdeen Group)*

Egy évvel ezelőtt azt mondtuk, hogy 2007-re „senkit sem fognak kirúgni azért, ha a Linuxot javasolja". Négy évvel csökkentve a magunk szabta határidőt most azt állítjuk, hogy egy informatikai beszerzőt manapság már ki is rúghatnak azért, ha nem veszi figyelembe a Linuxot. Ez kis lépés ugyan, de jelentőségében ahhoz hasonló, amit *Neil Armstrong* tett. *(EWeek)*

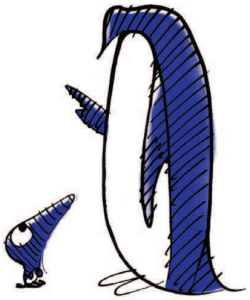
Az IBM termékcsaládoknak továbbra is támogatniuk kell mindenféle környezetet. Ilyen módon nem számít, hogy milyen téren működünk együtt az ügyféllel, a Linux mindenképpen a részét fogja alkotni. *(Jim Stallings – IBM Linux manager)*

Ezentúl nem vagyunk egyetlen fejlesztési környezetbe bezárva. Ha át akarnánk ültetni egy alkalmazást (HP-UX rendszerre), az gondot jelentene. A Linux használata azonban lehetővé teszi, hogy az alkalmazást általánosan elterjedt IBM-, HP- vagy Dell-gépen futtassuk, s közben kiaknázzuk a környezet által nyújtott előnyöket. *(Bridget O'Connor – Lehman Brothers)*

A Linux nem lesz túl hasznos az átlagembereknek – inkább a hozzánk hasonló vállalatok számára lesz az. *(Scott McNealy – CEO, Sun Microsystems)*

Linux Journal 2003. június, 110–111. szám

Egy informatikai beszerzőt manapság már ki is rúghatnak azért, ha nem veszi figyelembe a Linuxot.



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

A hónap szakmai tanácsai

Kickstart USB-s hajlékonylemezről?

A Red Hat 7.2 Kickstartját használom a Linux testreszabott változatának több gépre történő telepítésére. Jelenleg hajlékonylemezről indítom a rendszert, és a *ks.cfg* állományt a hajlékonylemezen tárolom, így a *syslinux.cfg* állományban van egy ilyen sor:

```
label ks
  kernel vmlinuz append text
  ➥ks=floppy initrd.img
  ➥lang=devfs=nomount
  ➥ramdisk_size=7168
```

Hogyan lehet elindítani a Kickstart-telepítést USB-s hajlékonylemez vagy USB-s CD-ROM használata esetén?

El tudom indítani a rendszert az USB-s hajlékonylemezről, illetve CD-ROM-ról, de az USB-eszközön lévő *ks.cfg* állományt a rendszer nem találja, ezért a Kickstart-telepítés sikertelen.

Vishali Karnik, Vishali.Karnik@respironics.com

A leírás szerint a Red Hat a 7.2-es változattól kezdve a telepítés alatt felismeri az USB-s hajlékonylemez-meghajtókat, így az elképzelés helyes. Nekem nincs USB-s hajlékonylemez-meghajtóm, így nem tudom kipróbálni, de a Fujitsu Siemens Computers segítőkész munkatársainak javaslata a ➲ http://www.fujitsu-siemens.com/partner/linux/readme/driver-disks-redhat.shtm alatt elérhető. A Linux az USB-s hajlékonylemez-meghajtókat SCSI-eszközként ismeri fel, ezért a *ks.cfg*-ben a `ks=floppy` bejegyzést `ks=hd:sda`-ra kell változtatni. Ha ez nem segít, a kézi telepítés során lépj be a héjba, és a */proc/scsi/scsi* állományban nézd meg, hogy milyen eszköznéven szerepel az USB-s hajlékonylemez-meghajtó. Az első eszköz a */proc/scsi/scsi* állományban az `sda`, a második az `sdb` stb.

Don Marti, dmarti@ssc.com

Nem szabványos modem használata

A Red Hat 8.0 nem ismeri fel Dell noteszgépem Pctel 2304 WT típusú modemjét.

Hari Babu Prasad, hari_bsnl@rediffmail.com

A PCtel alapú modemekhez Linux alatt illesztőprogram-modul szükséges. A nem hivatalos honlapon, a ➲ http://linmodems.technion.ac.il./pctel-linux címen megtalálható a legfrissebb változat, a támogatott modemek felsorolása és egy eléggé jó leírás.

Mario Bittencourt, mneto@argo.com.br

Levelezni szeretnék, de nincs POP-om!

Telepítettem egy internetkiszolgálót (Red Hat 7.3), de amikor egy másik számítógépről megpróbáltam a leveleimet POP3-mal elérni, a kapcsolatot visszautasította. A Sendmail programot ellenőriztem, fut.

Fausto Garcia, faustog@gesnet.com.mx

A levélkiszolgáló beállítása két részből tevődik össze. A Sendmail csak az egyik fele, az a levéltovábbító ügynök (MTA), ami az SMTP protokollt használja. Az SMTP kézbesít, „rátolja" a célrendszerre az üzenetet. Nincsen olyan szolgáltatása, ami az ügyfeleknek lehetővé tenné, hogy üzeneteiket „lehúzzák" a postaládájukba.

Chad Robinson, crobinson@rfgonline.com

A POP3 nem a Sendmail-csomag része a Red Hatban, hanem az IMAP csomagban található, egészen pontosan az imap-2001a-10 csomagban. A POP3 beállításához a következő lépések szükségesek:

1. A csomag letöltése az ➲ rpmfind.net/linux/redhat/7.3/en/os/i386/RedHat/RPMS/imap-2001a-10.i386.rpm címről;
2. az `rpm -Uvh imap-2001a-10.i386.rpm` paranccsal telepítjük;
3. a POP3-szolgáltatás engedélyezése a */etc/xinetd.d/ipop3* állomány szerkesztésével, amiben a *disable = yes* sort *disable = no* beállításra kell változtatni;
4. a szolgáltatás elindítása a `service ipop3 start` paranccsal;
5. a POP3 bejegyzése a rendszerindításkor elindítandó szolgáltatások közé: `chkconfig --level 345 ipop3 on`
6. a POP3-szolgáltatás kipróbálása.

Természetesen kipróbálni csak a kiszolgálón létező felhasználói azonosítóval lehet.

Felipe Barousse Boué, fbarousse@piensa.com

Az USB-billentyűzet rendszerindításkor leáll

Nemrég frissítettem Compaq Presario 7000-es számítógépemet Red Hat 7.1-ről Red Hat 8.0-ra. Az USB-billentyűzetem remekül működött a telepítés alatt. Miután az egység 3-as futási szintre vagy afölé lép, az USB-billentyűzet nem működik többé. Úgy kerülöm meg a hibát, hogy a *modules.conf* állományban a teljes USB-támogatást letiltom, ez azonban nem valami elegáns megoldás.

Doug Poulin, dougp25@yahoo.com

Győződj meg róla, hogy a rendszermag tartalmazza-e a *USB Human Interface Device (full HID) support* és a *HID input layer support* beállításokat. Add ki a `modprobe hid` parancsot, és nézd meg, hogy a modulban van-e egyáltalán. Ha igen, akkor add hozzá a következő sorokat a */etc/modules.conf* állományodhoz:

```
alias usb uhci
post-install uhci modprobe hid
```

Robert Connoy, rconnoy@penguincomputing.com

Modul betöltése rendszerindításkor

A Promise SuperTrack SX6000 RAID illesztőprogram-modul telepítésével kísérletezem Red Hat 7.3 alatt. Az illesztőprogramot lefordítottam, de tudom, hogy nem elég egyszerűen az `insmod` parancsot futtatni, ha azt szeretném, hogy a modul önműködően betöltődjön a rendszerindításkor. Van-e valamilyen példa vagy eljárás arra nézvést, hogy egy telepítőprogram milyen lépéseket hajt végre? Milyen beállítófájlok módosulnak, és hová kerül az illesztőprogram?

Steven Brown, sdbrown327@charter.net

Az illesztőprogram adatai a */etc/modules.conf* állományba kerülnek. Például az `alias eth0 eepro100` sor azt adja meg, hogy az `eepro100` illesztőprogramot az `eth0` ethernetcsatolóhoz kell betölteni. Ha a modul a rendszerindító eszközhöz kell, egy rendszerindító memórialemezre is szüksége lesz. Figyelmedbe ajánlom az `mkinitrd` súgóoldalát.
Robert Connoy, rconnoy@penguincomputing.com

A Red Hat összeállított egy leírást a rendszerindító parancsfájljainak a szerkezetéről, ami a ➲ http://www.redhat.com/support/resources/tips/Boot-Process-Tips/Boot-Process-Tips-3.html címen érhető el. A rendszerindításkor végrehajtandó saját parancsaidat is elhelyezheted itt.
Chad Robinson, crobinson@rfgonline.com

A ➲ http://www.linux.org/docs/ldp/howto/mini/Modules oldal áttekintést ad a betölthető modulokkal való munka alapjairól. A másik érdekes oldal, ami a Linux magmoduljairól szól, a ➲ http://www.luv.asn.au/overheads/kernelmodules címen található.
Felipe Barousse Boué, fbarousse@piensa.com

A Telnet védelmében?

Azt hiszem, *Don Marti* túl gyorsan elintézte a `telnet`-tel kapcsolatos kérdést (lásd A hónap szakmai tanácsait a Linuxvilág 2003. májusi számában). Az OpenSSH tényleg remek választás az `rlogin` és a `telnet` helyett a megbízható környezeten kívül. Egy megbízható hálózatban viszont az `rlogin`, a Telnet és a többi r-rel kezdődő parancs sokkal kényelmesebben használható az adott felhasználási területen. Nem szívesen nyugdíjaznám az `rlogin`-t és a Telnetet, különösen olyan helyen nem, ahol a Linuxnál régebbi rendszerekkel akad dolgom. Ahogy a biztonságnál mindig, a védelem árát (ebben az esetben a biztonságos eszköz bonyolultabb és kényelmetlenebb beállítását) az elérhető nyereséggel kell összehasonlítani. Szerintem az `rlogin` és a Telnet egyszerű lecserélése SSH-ra értelmetlen, ha a hálózat teljesen zárt és tűzfallal védett. Még egyszer, a megfelelő védelem kialakításához teljes körű elemzés szükséges.
Bob Hepple, bhepple@freeshell.org

A felhasználók hitelesítésére nyilvános kulcsokat alkalmazhatsz azokon a kiszolgálókon, amelyhez csatlakoznak. Használhatod az egyszerű jelszavas hitelesítést is. Mégha a jelszavas hitelesítést alkalmazod is, a jelszavakat védi az SSH szimmetrikus titkosítása. Napjainkban a vas újabb biztonsági kockázatot jelent. Biztos vagy benne, hogy az útválasztódon nincs telepítve egy trójai, amely a `telnet` csomagokat leszívja, és továbbítja őket a külső hálózatra? Biztos vagy benne, hogy a 802.11-es vezeték nélküli híd nem továbbítja a `telnet` csomagokat az éteren keresztül? Rengeteg gép érheti el az adataidat, és nem tudod ellenőrizni őket. Bob Hepple megállapítása, miszerint „az `rlogin` és a Telnet egyszerű lecserélése SSH-ra értelmetlen, ha a hálózat teljesen zárt és tűzfallal védett”, hibás. Mi mindig a biztonsági „léc felemeléséről” és újabb „hagymahéjak” beépítéséről beszélünk. Igaza van abban, hogyha a biztonság drága, akkor elemezni kell a fenyegetettséget, de az SSH annyira egyszerű, hogy az elemzésnek itt semmi értelme.
Christopher Wingert, cwingert@qualcomm.com

A korszerű terjesztések alapból tartalmazzák az OpenSSH-t, és külön erőfeszítéseket kell tennünk, ha Telnet-kiszolgálót szeretnénk használni. Szerencsére a kényelmes megoldás és a biztonságos megoldás ugyanaz. A SSH a jelszavas hitelesítéssel ugyanolyan egyszerű, mint a Telnet. Megfelelően beállított kulcspárokkal még annál is egyszerűbb. Ha más operációs rendszer alól kell bejelentkezned Linux-rendszerbe, nézd meg *Rick Moen* SSH-ügyféllistáját a ➲ linuxmafia.com/pub/linux/security/ címen
Don Marti, dmarti@ssc.com

Frissítés után eltűntek a nyomtatók

Nemrég frissítettem StarOffice 5.2-ről 6.0-ra, de a rendszerem nem találja a hálózati nyomtatókat. A nyomtatók HP LaserJetek JetDirect kártyákkal, és mindegyiknek van saját IP-címe. Az `spadmin` csak az alapértelmezett nyomtatót ajánlja fel, nincs egyszerű módja az IP-cím megadásának. A *psprint.config* állományban lehetőség nyílik a helyérték beállítására, de az IP-nyomtató megadása nincs leírva.
Murray Zangen, murray@nj.com

Először is a `printtool` eszközzel add hozzá a rendszerhez a nyomtatókat, és add meg, hogy JetDirect nyomtatókról van szó. Ezután az `spadmin` eszközben add hozzá a nyomtatót a StarOffice-hoz. Amikor a nyomtatási parancs ablakhoz érsz, add meg a nyomtatási parancsot. Például:

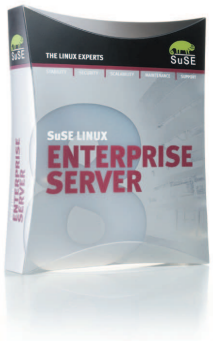
```
lpr -P hp3
```

Keith Trollope, ktrollope@san.rr.com

Memóriateszt Linux alatt

Szinte minden Linux-változathoz elérhető a memtest86 program. Debian/GNU Linux alatt az `apt-get install memtest86` paranccsal telepíthetjük a legegyszerűbben, ekkor a */boot* könyvtárba települ egy *memtest86.bin* nevű fájl. Ahhoz hogy a memóriát tesztelhessük, a rendszerindításkor erre a fájlra kell hivatkoznunk. Akár Grub, akár LILO a rendszerbetöltőnk, beállítási példákat találhatunk a */usr/share/doc/memtest86/examples* könyvtárban. Ha nem akarjuk a rendszerindító programunkat birizgálni, akkor helyezzünk be egy hajlékonylemezt a meghajtóba, és a következő parancsot rendszergazdaként adjuk ki: `dd if=/boot/memtest86.bin of=/dev/fd0`. Ezután a gépet a hajlékonylemezről indítva azonnal elindul a memtest86 program.
Csontos Gyula, csontos@linuxvilag.hu

Új termékek



SuSE Enterprise Server 8 AMD64-re

Az AMD 64 bites Opteron processzorát támogatva megjelent a United-Linux 1.0-n alapuló SuSE Linux Enterprise Server 8. A Server 8 kiszolgáló operációs rendszer kihasználja az Opteronnak azt a képességét, hogy mind 32 bites, mind 64 bites környezetben képes alkalmazásokat futtatni. A Server 8-ban javított 2.4.19-es rendszermag (beleértve a GCC 3.2.2-t) van, ami támogatja a magas rendelkezésre állást és a nagy teljesítményű kapcsolatot a háttértárolókkal az aszinkron B/K, többutas memóriaelérés és akár 600 merevlemez kezelése lehetőségének segítségével. A Server 8 64 processzorig és 512 GB memóriáig méretezhető.

Adatok: ULX Kft. SuSE Linux AG magyarországi irodája
telefon: (1) 450-0921,
fax: (1) 330-5672,
támogatás: (1) 236-0053,
➲ http://www.suselinux.hu

HyperBlade Opteron géptelep, 1U és 2U kiszolgálók

Az APPRO bejelentette új HyperBlade Server Cluster nevű termékeit, amelyek 1U és 2U magas két AMD Opteron processzort tartalmazó kiszolgálók. Az APPRO HyperBlade fürtöt a nagy számításigényű piaci szereplők elvárásainak megfelelően tervezték. A HyperBlade Server Cluster tömeggyártásban előállított x86-alkatrészekből álló nagy sűrűségű fürt. Akár 80 számítógépkártya is elhelyezhető benne, ami 160 AMD Opteron processzort jelent. Az APPRO 1U és 2U kétprocesszoros kiszolgálói nagy processzor- és memória-sávszélességgel bírnak. Főbb jellemzőik: két AMD Opteron processzor, 32 és 64 bites számítások egy időben, HyperTransport, PCI-X támogatása, cserélhető merevlemezek és legfeljebb 16 GB DDR SDRAM.

Adatok: APPRO, 446 South Abbott Avenue, Milpitas, California 95035,
➲ http://www.appro.com

Mandrake Corporate Server 2.1 Opteronra

Az AMD Opteron processzorának megjelenésével párhuzamosan a MandrakeSoft Inc. kiadta a Mandrake Linux Corporate Server 2.1-et. A Server 2.1-et nagy teljesítményigényű alkalmazásokhoz, például webkiszolgálókhoz, adatbázis-kezelőkhöz (MySQL-64), alkalmazás- és állománykiszolgálói feladatokhoz tervezték. A felhasználók úgy térhetnek át a 64 bites alkalmazások futtatására, hogy a régi 32 bites alkalmazásokat is tovább használhatják. A Server 2.1 a 2.4-es rendszermagon alapul, és a következőket támogatja: DHCP, Apache, Postfix és Squid proxykiszolgáló; POP3, IMAP és webes levelezőszolgáltatások; valamint többféle naplózó fájlrendszer.

Adatok: MandrakeSoft, Inc.,
2400 North Lincoln Avenue,
Altadena, California 91001,
➲ http://www.mandrakesoft.com

Little Board 700

Az Ampro Computers Inc. Little Board 700 nevű egykártyás számítógépét az igényes, teljesítményérzékeny beágyazott alkalmazások számára tervezték. A Little Board 700 beágyazott számítógép EBX alakú (5,75" × 8"). Többféle alacsony feszültségű processzorral kapható, például ilyen a 933 MHz-es Pentium III 512 KB gyorsítótárral, a 650 MHz-es Celeron és a 400 MHz-es Celeron. Legfeljebb 1 GB DRAM-ot támogat, és van beépített hőmérséklet-figyelő és energiaellátás-kezelő rendszere. Az alaplapra épített Type II CompactFlash aljzat legfeljebb 1 GB flashmemóriát képes fogadni IDE-merevlemezként felismerve. Az alaplapon még a következő eszközök találhatók: két 10/100-BaseT ethernetvezérlő, AGP 4x videovezérlő a TFT megjelenítők támogatásával, AC97 hang, két USB-kapu, négy soros kapu és PC/104-Plus sín.

Adatok: Ampro Computers, Inc.,
5215 Hellyer Avenue #110, San Jose, California 95138,
e-mail: info@ampro.com,
➲ http://www.ampro.com

Mini-Box M-100

A Mini-Box beágyazott és általános célú számítógép-alkalmazások építésére tervezett x86-os számítógép. A Mini-Box a VIA Mini-ITX alaplapjára épül és 12 V-ot használ, ezáltal alkalmas olyan területeken való felhasználásra, ahol lényeges az

alacsony fogyasztás és a zajtalan működés. A Mini-Box LCD-vel és testreszabható 14 gombos billentyűzettel van felszerelve, ami kiváltja az egeret és a billentyűzetet. A Mini-Box súlya nagyjából egy kilogramm, az alapfelszereltséghez hozzátartozik a 256 MB PC133-as RAM, 64 MB CompactFlash és 800 MHz-es x86 VIA C3 processzor. Választható többek közt 533 MHz-es ventilátor nélküli processzor, 128 MB CompactFlash és 40 GB-os 2,5"-os IBM merevlemez. A Mini-Box kisméretű beágyazott Linuxot, de teljes szabványos terjesztést is futtathat.

Adatok: Ituner Networks Corp.,
3071 Southwycke Terrace,
Fremont, California 94536,
➲ http://www.mini-box.com

Az Aspen Systems Opteron Beowulf-telepei

Az Aspen Systems Inc. az AMD-vel együttműködve bejelentette testreszabott, Opteronra alapuló géptelepeit és csúcsszintű kiszolgálóit. A géptelepek és kiszolgálók jellemzői: 1U, 2U és 4U magas, keretbe szerelhető kivitel nagysebességű Myrinet eszközökkel összekötve; Aspen Beowulf Cluster Management program; AMD 64 bites processzorok, a régi 32 bites és az új 64 bites alkalmazásokhoz; HyperTransport az áramkörök közötti gyorsabb adatcseréhez; valamint 128 bites DDR DRAM-vezérlő, ami processzoronként legfeljebb nyolc DDR DIMM memóriamodult képes vezérelni.

Adatok: Aspen Systems, Inc., 3900 Youngfield Street, Wheat Ridge, Colorado 80033,
➲ http://www.aspsys.com

Linux Journal 2003, 111. szám

Az ingyen sört nehéz eladni?

Azokban az országokban, ahol az olcsó, törvénytelen másolatokkal kell versenyezni, az átláthatóság és a testreszabhatóság az igazi érv a Linux mellett.

A nyílt forrású programok testreszabhatók, bővíthetők, és olcsón vagy ingyen beszerezhetők. A zárt kódú programok nehezen bővíthetők vagy alakíthatók át, és gyakran hihetetlenül drágák. Ennek következtében a nyílt forrású megoldásokat világszerte széles körben alkalmazni kezdik a vállalkozások és a kormányok egyaránt, sőt néha egészen odáig elmennek, hogy törvényben írják elő a használatukat. Azt gondolhatnánk, hogy a fejlődő országok járnak élen a szabadon hozzáférhető, kiváló minőségű programok bevezetésében – de tévedünk.

A Geekcorps több száz, fejlődő országokban folyó informatikai programoknak a támogatása során gyűjtött tapasztalatai alapján úgy látjuk, hogy a nyílt kódot meglepően nehéz eladni. A legfőbb hiba a fejlődő országokban az „ingyen sör" túlhangsúlyozása. Ez ***Richard Stallman*** szólásmódja, szerinte „a program legyen szabad (free), abban az értelemben, ahogy a szólásszabadság (free speech), nem úgy, ahogyan az ingyen sör (free beer) kifejezésben szerepel". A nyílt kódú programok alacsony költségére hivatkozni gyakran egy öngóllal ér fel. A műszaki megoldások átadása és az ehhez kapcsolódó programok története során gyakran próbáltak elavult megoldásokat rásózni a fejlődő országokra, amikor ezeket a fejlett országokban már nem lehetett eladni. Így aztán az alacsony költség és az alacsony színvonal közé sokan tévesen egyenlőségjelet tesznek.

A helyzet az, hogy egy szabad program a fejlődő országokban gyakran többe kerül a felhasználóknak, mint egy jogdíjas program. A szerzői jogok megsértése széles körben elharapódzott, így aztán a programokat gyakran az adathordozó áráért meg lehet kapni. Örményországban, Jerevánban nemrég láttam a Microsoft Windows XP és a Red Hat 8 példányait egy üzletben, mind a kettőt öt dollár alatti áron. A Windowst ismerő felhasználóknak a Linux többletköltséget jelent – a program használatához nélkülözhetetlen leírások költségét. Bár az elektronikus formában elérhető leírások megoldást jelenthetnek a gondok egy részére, ezek beszerzése is lehet költséges vagy riasztó. A 3 MB-os Emacs kézikönyv letöltése kezd drága mulatság lenni, ha az ember percdíjat fizet egy olyan netcafében, ahol húsz gép osztozik egy 28,8 kb/s sebességű modemen. Külföldi szakembereknek kérdéseket föltenni egy nem túl jól ismert nyelven még ennél is riasztóbb élménnyé válhat, amikor a kezdők először találkoznak a netikett szabályaival. A jó szándékú „RTFM" választ könnyen félreértheti olyasvalaki, aki egy, az udvariasságra és formaságokra nagy hangsúlyt fektető kultúrából érkezik. A világ szinte összes kultúrája udvariasabb, mint az infós szakiké.

Noha a nyílt forrás népszerűsítőinek figyelembe kell venniük a nyelvi, kulturális és sávszélességbeli korlátokat, amik a fejlődő országokban befolyásolják a szabad programok terjedését, leginkább arra kell összpontosítanunk, hogy megmutassuk: a nyílt kódú programok fejlettebbek és hatékonyabbak lehetnek vetélytársaiknál. Ez azt jelenti, hogy a nyílt kód szólásszabadság-jellegű előnyeire kell összpontosítanunk: a bővíthetőségre, átláthatóságra és az így létrejövő magas használati értékre. A „szólásszabadság" szellemében végzett fejlesztések egyik kitűnő példája a ➲ http://translate.org.za kezdeményezés, aminek az a célja, hogy operációs rendszereket és kulcsfontosságú programokat tegyen hozzáférhetővé Dél-Afrika 11 hivatalos nyelvén. Munkájuknak – és a Mozilla programkód elérhetőségének – köszönhetően már létezik böngészőprogram zulu, xhosa és még négy másik nyelven is. Az IDN, Ghána egyik vezető internetszolgáltatója (➲ http://www.idngh.com), a SAVA vezeték nélküli hozzáférési pontok segítségével önti új formába üzleti tevékenységét. A SAVA egy olyan, Ghánában gyártott linuxos gép, ami WiFi hozzáférést tesz lehetővé vállalkozások és netcafék számára, kikerülve a fejletlen ghánai telefonhálózatot. A szolgáltatók az egész földrészen figyelemmel kísérik az IDN sikerét, és nem egy cseréli le a rendszereit Linux-kiszolgálókra.

Ahhoz, hogy a nyílt forráskód valóban világszerte jó fogadtatásra találjon, a fejlesztőknek a fejlődő országokra jellemző informatikai nehézségekre kell összpontosítaniuk. Az olyan kezdeményezés, mint a Simputer (➲ http://www.simputer.org) – egy olcsó, Linux alapú kézi gép Indiában – ennek az egyik lehetséges modelljét jelenti: a világméretű közösség támogatja egy fejlődő ország tehetséges mérnökeinek egy csoportját. Az MIT Media Lab ThinkCycle kezdeményezése (➲ http://www.thinkcycle.org) egy másik modellre mutat példát: a világ különböző egyetemein tanuló tervezőket párosít össze a fejlődő országokban működő nonprofit szervezeteknél fölmerülő mérnöki feladatokkal. Az amerikai kormány Digitális Szabadság Kezdeményezése ➲ http://www.digitalfreedom.gov egy újabb modellt mutat be, problémamegoldó csoportokat hozva létre szenegáli fejlesztőkből, amerikai és szenegáli felhasználók közül kikerülő üzleti és informatikai szakértőkből.

A politikai fejlemények is elősegíthetik a nyílt forrás világméretű terjedését. Egyre több politikus szorgalmazza, a perui ***Edgar Villanueva Nu□ez***-hez hasonlóan, hogy a nyílt forrás használatát törvény írja elő az információ „sértetlenségének, bizalmas kezelésének és hozzáférhetőségének" védelme érdekében.

A fejlődő országokban a nyílt forráskód népszerűsítői más irányban is tevékenykedhetnek. Követelhetjük, hogy országunk külföldi segélyeket elosztó szervezete a pénzt a fejlődő országokban olyan új informatikai fejlesztésekre költse, amelyekkel nyílt kódot hoznak létre.

Linux Journal 2003. július, 111. szám

Ethan Zukkerman
(ethan@geekcorps.org)
A Geekcorps Edgar Villanueva Nu□ez nevű nonprofit önkéntes szervezet alapítója.
➲ http://www.geekcorps.org

Hogyan teszi a Linux okosabbá a vállalatokat? (1. rész)

A forgalmazók „megoldásaikról” szóló meséi nagyon szépek, de hogyan változtatja meg a Linux a hagyományos vállalatok információtechnológiáját?

A vállalatok Linux-sikereinek két nézőpontja létezik: az egyik egy belső, a másik pedig egy külső. A beavatottak a szervezet intelligenciáját, leleményességét és szakértelmét tartják nagyra. A külső szemlélők azokat a forgalmazókat becsülik, akik ugyanezekkel az erényekkel bírnak. Az érdem természetesen mindkettőjüké, és bizonyos mértékig mind két fél elismeri a másik teljesítményét. A sajtó azonban általában külső nézőpontból szemlélődik, és az érdemek oroszlánrészét a forgalmazóknak tulajdonítja. Ennek eredményeképpen nagyon csekély elismerést érdemelnek azok az értékek, amelyek valóban hozzájárulnak ahhoz, hogy a Linuxot sikeresen alkalmazzák a vállalatokon belül.

Belülről nézve

Nincs két egyforma belső történet. Minden forgalmazói történet alapvetően arról szól, hogyan oldották meg a termékek az adott feladatot. Ezért a forgalmazók szeretik termékeiket „megoldásnak” nevezni. A Linux azonban nem a „megoldásokról” szól – ez a forgalmazók szóhasználata. A Linux a leleményességről és az értelemről szól – ezek a vevők kifejezései. A Linux úgy válik sikeressé, hogy az ügyes vállalatoknak segít abban, hogy még ügyesebbé tegyék magukat. A Linuxszal és a nyílt forrással, a piac elsődleges kínálati és keresleti oldala is a vállalaton belül van jelen.

Ezt az együttélést sokunknak nagyon nehéz megértenie. Nehéz a forgalmazóknak, mert olyan piachoz vannak szokva, ahol a kínálati oldal az ellenőrzésük alatt áll, és a nagy haszonkulcs a nagyszerű, még akkor is, ha a termék már nem is szabvány. Nehéz megértenie a sajtónak is, mivel a legtöbb általuk megírt hír egy termék–termék közti sportközvetítés. Ebben a játszmában a piac a játéktér, a vevők pedig a megnyerhető nyeremények. Nehéz megérteniük a vállalati vezetőknek is, mivel annyira hozzászoktak ahhoz, hogy a nyeremény szerepét játsszák a kereskedők döntő csatáiban.

Akiknek nem esik nehezükre ezt felfogni, azok az emberek, akik a Linuxot azért használják, hogy a vállalataikat eredményesebbé, megbízhatóbbá és hatékonyabbá tegyék.

A vevő diktál

Nézzük az LSI Logic, a félvezetőgyár példáját. ***Roland Smith***, az általános operatív igazgató vezeti annak a négy csoportnak egyikét, amelyik közvetlenül a főinformatikusnak jelent. A vállalat Linuxra való nagy váltása azután történt, hogy Smith csoportja a HP-UX Linuxra való cserélését javasolta. A teljesítmény ezáltal 5–6 szorosára növekedett, a költségek viszont kétharmadnyival csökkentek. Smith a következőket mondja: „Bárki, aki betér hozzánk, és a főinformatikusunkkal egy új alkalmazásról, projektről vagy bármilyen más új dologról szeretne beszélni, a vezető informatikus első kérdése az lesz: „Működik Linux alatt is?” Ennek a környezetnek akad néhány érdekes hatása a támogatás folyamatára. Smith egy kiváló példát mesél el a forgalmazó és a vevő közötti terméktámogatásról: „Nemrégen elhatároztuk, hogy SAP-rendszerünkhöz beállítunk néhány Linux alkalmazáskiszolgálót. Kimentünk a piacra, beszéltünk a Dell-lel és a HP-val. Elmondtuk nekik, hogy mit szeretnénk. Ez egy tesztkiszolgáló volt, ami szerintük a legalkalmasabb Linux futtatására. Négy processzort és 1 GB memóriát kértünk.

Ekkor történt néhány érdekes esemény. A HP körülbelül két hétig össze-vissza hadonászott, mígnem visszajött és megkérdezte: Mire is van tulajdonképpen szükségük? A Dellnek három napjába került, hogy hozzon egy kiszolgálót. Ők ezt mondták: Tessék, három hónapra rendelkezésükre bocsátjuk a kiszolgálót. Kérem, meséljék majd el, mire jutottak vele.

Ez szinte csoda volt. Majd elmentünk az SAP-hez és azt mondtuk nekik: Mit ajánlanak? Az SAP-nél a következőket válaszolták: Nos, úgy gondoljuk, SuSE 8 alatt jól fut. Mi pedig úgy véltük, hogy Red Hat 7.1 alatt fut jól, de nem voltunk benne igazán biztosak. Ezután elmentünk a Red Hathez, ahol ezt felelték: Úgy hisszük, hogy az Advanced Server 2.1 változatunkon jól fut, de nem igazán tudjuk. Mi lenne, ha Önök kipróbálnák, és elmondanák nekünk a tapasztalataikat?

Így megépítettük a rendszert. Megvettük a Red Hat alkalmazáskiszolgálót, megépítettük, telepítettük rajta az SAP-t, és betettük a próbakörnyezetünkbe. Elképesztően jól futott, ennek nagyon örültünk. Ami azonban a legérdekesebb a számomra, hogy az összes forgalmazó cég közül csak a Dellnél mondták, hogy nagy örömmel biztosítják a gépet, hogy kipróbálhassuk, de többet sajnos nem tudnak hozzátenni a dologhoz.”

Az LSI ágazatában az irányítás legjava a vevői oldalon található.

Versenyelőny

Hasonlóan ügyes vállalat az Orbitz, online utazási iroda. ***Leon Chism***, vezető internetmérnök a következőket nyilatkozta: „Fejlesztőink nagyon ügyesek a nyílt forráskódú alkalmazások használatában. Céljuk, hogy javítsák cégünk programtervezési, -fejlesztési és -vezérlő eljárásait. Saját ötleteik nyomán fejlesztettek ki számos eszközt, ami naplófájlokat elemez, PostgreSQL adatbázisban tárolja őket, majd ex post facto elemzést készít a hibák feltárása, valamint a lehetőségek kiaknázása érdekében. Az első ilyen eszköz, amiről hallottam, egy olyan program volt, ami többször is sikeresen elemezte és javította ki a kód néhány részletét. Számos fejlesztőnk és mérnökünk akad, akik olyan eszközöket használnak, mint a Perl, Python és Jython (egy hivatásos Jython-fejlesztőt-alkalmazunk). Ezeket saját alkalmi eszközeikhez használják, amelyek a webhely-viselkedés ellenőrzésére szolgálnak. Mindez plusz a hivatalos termékirányítás háttere mellett. Általában, miután kifejlesztettek egy eszközt, kipróbálták és hasznosnak bizonyult, a vezetés az informatikai háttér részévé teszi, s ezt azután a hálózati működési központunk használja.”

„Az Orbitz a Linuxot és a nyílt forrású programokat versenyelőnynek tekinti: az Orbitz olyan vállalat, amely termelési környezetben alkalmazza a nyílt forrású projekteket, és nem fél ezeket nyilvánosságra is hozni. Jelenleg több mint 750 linuxos számítógépet használunk a termelésben. Webkiszolgálóink Linuxon futó Apache programok, alkalmazáskiszolgálóink Linuxon futó kereskedelmi servletmotorok, a háttér „könyvelőmotor"-t pedig kizárólag Java-szolgáltatások alkotják, amelyek szintén Linux alatt futnak. Végül az alacsony díjakat kereső programunk, ami az egyik legfőbb megkülönböztetőnk versenytársainkkal szemben, szintén Linux alatt fut."
Az LSI és az Orbitz példáján keresztül nagyon jól láthatjuk, hogyan alkalmazzák az ügyes vállalatok a Linuxot és a nyílt forrású programokat mind a vállalaton belül, mind pedig a vállalat és a külső forgalmazó kapcsolatában. Most nézzük, mi a helyzet a külső nézőponttal!

Külső nézőpontból

Kitűnő példát szolgáltat számunkra ***William M. Bulkeley*** „Kilépve árnyékából: a nyílt forrású program nemcsak elfogadottá válik, de egyben óriási üzlet" című cikkben, mely a Wall Street Journal 2003. március 31-i számban jelent meg.
„A vállalatok megtalálták annak módját, hogy pénz csináljanak a nyílt forrású programok felhasználóinak támogatásából, vagy abból, hogy az általuk árusított programhoz hozzáadják ezeket az ingyenes programokat. A profitmotiváltság miatt a szabad program egyre több helyen bukkan fel a vállalati és a kormányzati számítógépeken."
Amikor először megláttam, azt gondoltam: nagyszerű, később azonban feltettem magamban néhány kérdést is:

- Miért fontos az, hogy a vállalatok „megtalálták annak módját, hogy pénzt csináljanak"? Fontosabb annál a munkánál, amit megoldanak vele?
- Miért mondja azt, hogy „vállalatok", amikor a „forgalmazókra" gondol? A vevők már nem is vállalatok?

Nézzünk szembe a tényekkel: a Linux-felhasználók kemény vásárlók. Újra Leon Chismet idézem: „A kapcsolatok frontján akad egy forgalmazónk, aki még mindig egy nyílt forrású megoldás bevezetésén gondolkozik. Az alkalmazás legtöbb része működik, de amikor hibát találunk benne, a nehézség megoldása sok kívánnivalót hagy maga után.
Az eszkalációs eljárások, a root cause, hosszú viták a „támogatott kiépítésekről" mind-mind az eljárás elmocsarasadásához vezetnek a megoldásra való összpontosítás helyett.
Ha a Linuxról vagy az Apache-ról van kérdésünk, akkor ezek a dolgok már nem kérdések. Egyszerűen alkalmazzuk a támogatásprogramot, mely programok mind a ➲ http://www.google.com-nál végződnek, és megkezdjük a keresést.
A másik lehetőség, hogy elkezdjük figyelmesen áttanulmányozni a forráskódot. Az Orbitz olyan vevő, amelyik a legtöbbet hozza ki az általa használt programból. Néhány forgalmazó megfelel a kihívásnak, és erőfeszítéseket tesz a vevői igények támogatására. Néhányan azonban nem hajlandóak erre.
Mi pedig nem tudjuk, ki kicsoda, mindaddig, ameddig – ahogy ők mondják – nem jön egy lökés. A nyílt forrású termékek használata azonban megoldja ezt a kérdést. Felállítottunk egy programmérnöki és fejlesztési csapatot, akik képesek a saját hajójukat mozgásban tartani, és a program szerzőitől, levelezőlistákról és a Google-tól kapott támogatás a legtöbb esetben több mint elég."
Ma mégis a kemény forgalmazó játékát játsszuk, miközben a kemény vevő története nem működik.

Az egyre sürgetőbb értelem

Lehet, hogy a Dell a legnagyobb példányszámban eladott PC-márka, de az igazi közönségsikerek a nevenincs „fehér" dobozok, amelyeket igény szerint szerelnek össze szabványos alkatrészekből. A vásárlók azért kedvelik őket, mert olcsók és könnyen szerelhetők. Ezekről azonban nem sokat hallunk. Ennek egyik oka, hogy a gyártóik nem költenek sokat reklámra; a másik ok az, hogy a számítógépiparnak régóta meglévő előítélete van a „árucikk" szóval szemben, és rettenetesen fél az „árucikké válástól".
A programiparnak azonban szembe kell néznie azzal a ténnyel, hogy a vásárlók szeretik az árucikkeket. Nagyon jó példa erre a Boulderben, Colorado államban székelő Légkörkutatási Egyetem (University Corporation for Atmospheric Research – UCAR, ➲ http://www.ucar.edu/ucar/index.html). Neve ellenére az UCAR egy vállalkozás, ami tekintélyes vevőkörrel bír, ilyen például a nemzeti légi közlekedési felügyelet. ***Greg Thompson***, az UCAR egyik tudósa elmagyarázza, hogyan zajlik a döntéshozatali eljárás cégüknél: „Itt mindenkinek van egy polca, ami tele van O'Reilly-könyvekkel. Szeretjük, ha magunk találjuk ki a dolgokat. Példának okáért én tudós vagyok, nem pedig programozó. Nem olyan rég azonban arra volt szükségem, hogy textfájlok és a Perl segítségével adatokat kezeljek. Így elkezdtem olvasni az adatbázisokról, és megtanultam a MySQL-t. Nem érdekelt volna a dolog, ha a MySQL nem lett volna ingyenes. Nekünk nincs ötszámjegyű költségvetésünk az Oracle-re. Nincs szükségünk mély adatbázisos dolgokra. Nincs szükségünk tranzakciókövetésre. Olyan eredményekre van szükségünk, amelyek a Weben keresztül elérhetők, ezért a MySQL ésszerű választás volt. A MySQL körüli belső szakértelmet egyszerűen ki lehet alakítani. Végzek egy kis kutatómunkát, megnézem, hogy ami szükséges, az telepítve van-e, ha nem, küldök egy segélykérő elektronikus levelet egy rendszergazdának. Majd a következő napon letöltöm a Debian-csomagot, futtatom az `apt-get` parancsot, és már működik is. Mindez egyetlen fillér elköltése nélkül."
Elképesztő, hogy milyen kicsire zsugorodik a programengedélyezési tortúra. Ha kicsi a feszültség és a költség nulla, a szabad vállalat méltó a nevéhez. „IT-költségvetésünk nulla", nyilatkozta ***Elliot Noss***, a torontói Tucows (tucows.com) elnöke és ügyvezető igazgatója. A Tucows a világ egyik legnagyobb programletöltő webhelyének üzemeltetésével foglalkozik. Ezen felül a három elsőszintű tartománynév-bejegyzők egyikét is üzemelteti (OpenSRS, ➲ http://resellers.tucows.com/opensrs). Körülbelül 3,5 millió tartománynév karbantartásával foglalkoznak, mindezt Linuxon, Apache, MySQL és PostgreSQL segítségével. „El sem tudom képzelni, mennyibe is kerülne, ha egy nagy Sun-rendszeren vagy egy óriási Oracle-adatbázison futtatnánk az egészet" – nyilatkozta Noss.
A technológiai és a hálózattal foglalkozó cégek sokkal inkább hajlandók arról beszélni, hogy Linuxot és nyílt forrású rendszereket használnak. Más típusú cégek azonban csendesen változtatják meg programkezelésük folyamatát. Írásunk következő részében néhány gyakorlati példával ismerkedünk meg.

Linux Journal 2003. július, 111. szám

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

Páros X: Linux és Xbox

Ha módosítasz egy Xboxot, értékes ismeretekre tehetsz szert, miközben egy tetszetős kis gépecskéhez is hozzájutsz.

A Microsoft 2001 novemberében az Xbox által belépett a videokonzolok piacára egy olyan géppel, amely azóta is felülmúlja minden vetélytársát, ha a processzor sebességét és a megjelenítési teljesítményt vizsgáljuk. A SEGA Dreamcast gépéhez hasonlóan az érdeklődő szakemberek 2002 májusában kezdték meg a Linux Xboxra való átültetését. Mindössze három hónap telt el, és már meg is jelentek a világhálón egy Xboxon futó Linux-rendszermag első üzenetei. Most, egy évvel a tervezet indulása után, a Linux az Xbox összes változatán megbízhatóan fut, az Xbox Linux pedig készen áll a mindennapos használatra.

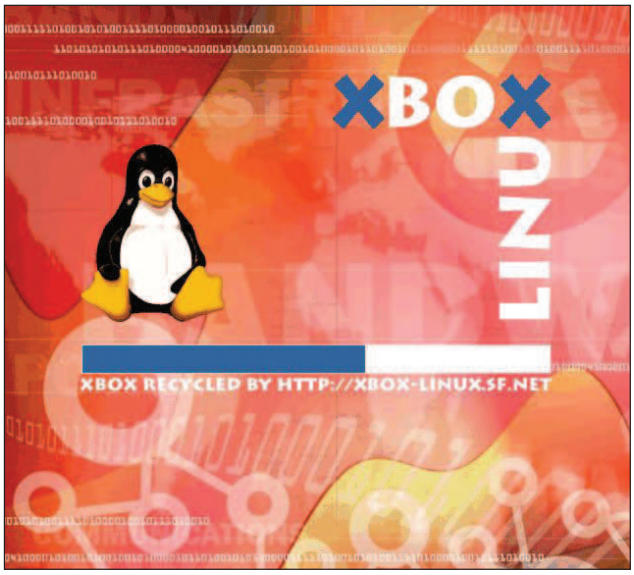
Az Xbox felépítése

Az Xbox belsejében egy 733 MHz órajelű Intel Celeron processzort, 64 MB DDR RAM-ot (a videokártyával megosztott), egy nVidia GeForce3 grafikus feldolgozóegységet (GPU), egy 8 vagy 10 GB-os merevlemezt, DVD-ROM-meghajtót és hálózati kártyát találunk. A vezérlőket négy USB jellegű aljzatba csatlakoztathatjuk, illetve még egy tévékimenet is rendelkezésünkre áll (*1. ábra*). Az eszközök listája alapján inkább egy elfogadható teljesítményű asztali gépre gondolhatnánk, nem egy játékkonzolra. Az Xbox esetében nem csupán arról van szó, hogy a belsejében szabványos PC-s alkatrészeket találunk – Intel processzor, nVidia grafikus lapka –, hanem ténylegesen egy kisméretű, fekete házba szerelt személyi számítógépről, amit kisebb módosításoknak vetettek alá. Az Xbox lapkakészlet két összetevője az NV2A Northbridge és az MCPX Southbridge, mindkettőt az nVidia fejlesztette. Az nVidia nForce lapkakészlet, amit PC-kben is megtalálunk, lényegében megegyezik az Xbox lapkakészletével. A Southbridge lapka MCP feliratot kapott, tudása az MCPX-ével megegyező: USB-vezérlőket, egy IDE-vezérlőt, ethernetcsatolót és egy AC97-megfelelő Dolby Digital hangvezérlőt foglal magába.

Az Xbox háttere roppant egyszerű. Mivel a Microsoft PC-re már rendelkezett saját operációs rendszerrel, rendszer- és DirectX könyvtárakkal, az Xbox tervezésekor is e jól bevált géptípus mellett döntöttek. A Microsoft eleinte AMD processzort és lapkakészletet akart szerelni az Xboxba, a videolapka eredetileg is nVidia lett volna. Később módosítottak az elképzeléseken, így került bele Intel processzor. Az nVidia tehát megvette a lapkakészletet az AMD-től, legyártotta a lapkákat az Xboxokba, majd ugyanezt a dolgot nForce néven a PC-s piacon is megjelentette.

Az Xbox és a PC közötti hasonlóság nemcsak a Linux telepítését és futtatását könnyítette meg jelentős mértékben, de meg is adta mindennek az értelmét: az Xbox egyszerű számítógépként is remekül használható. A Dreamcast, a PlayStation 2 vagy GameCube nem rendelkezik merevlemezzel és ethernetcsatolóval, az Xbox viszont igen. A szabványos PC-s eszközöknek köszönhetően kisebb módosítások után tehát a Linux-terjesztések is futtathatók Xboxon.

Kedvező ára és kis mérete révén egy Linuxot futtató Xbox egy normál PC-t helyettesítve nagyszerűen használható asztali



1. kép Linux indulása az Xboxon

számítógépként (*2. kép*), vagy akár kiszolgálóként, tévékimenete révén pedig szórakoztatóeszközként is alkalmazható, például filmek lejátszásához vagy zenehallgatáshoz.

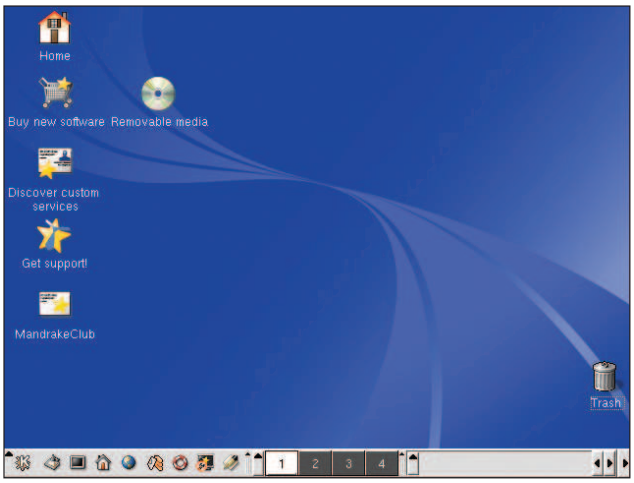
Biztonság

Igaz, hogy az Xbox nagyon hasonlít egy szabványos PC-re, ám a Linux telepítése mégsem intézhető el annyival, hogy behelyezünk egy telepítő CD-t. Először is az Xbox rendszerindítási folyamata nagyban eltér a PC-kben megszokottól. A normál számítógépekben PCBIOS (alapszintű B/K rendszer) található egy ROM-ban, ami 16 bites könyvtári eljárásokat tartalmaz a billentyűzet, a megjelenítő és a merevlemez kezelésére, illetve egy egyszerű rendszertöltő is helyet kapott benne, ami beolvassa valamelyik tárolóeszköz első szektorát, majd futtatni kezdi annak tartalmát. Az Xbox nem rendelkezik ilyen BIOS-szal. Egy 256 KB-os ROM-lenyomatot találunk benne, ez egy statikusan befűzött, lecsupaszított, Windows 2000 alapú rendszermagot tartalmaz, aminek a futtatása az Xbox bekapcsolásakor azonnal megkezdődik. A merevlemez – amely egyébként egyedi ATA jelszóval védett, így más számítógépbe helyezve nem olvasható, illetve másik lemezzel nem helyettesíthető – semmilyen operációsrendszer-összetevőt nem tárol. Az Xbox rendszermagja indításkor feloldja a merevlemezt, majd CD-ről vagy DVD-ről megpróbálja futtatni a ***default.xbe*** fájlt. Ha ilyet nem talál, akkor az ***xboxdash.xbe*** fájl futtatásával próbálkozik a merevlemezről. Ez lényegében rendszerbeállításokat és egy zenei CD-lejátszó alkalmazást foglal magába, tárolása állandó jelleggel a merevlemezen történik.

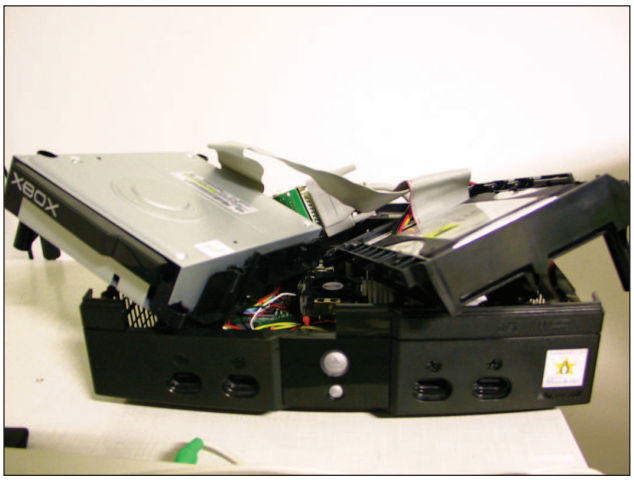
A ***.xbe***-fájlok futtatható állományok, nagymértékben hasonlítanak a Linux ELF-fájlokra, azzal a kivétellel, hogy a Microsoft

```
.Xbox
|-- Intel Pentium III Celeron Coppermine CPU
|-- 64 MB RAM
|-- Flash ROM
`-- PCI bus
    |-- Programmable Interrupt Controller
    |-- System Timer
    |-- DMA Controller
    |-- PCI Bridge Device - Host Bridge                    (0:0:0, 0x2a5)
    |-- Memory Controller - SDRAM                          (0:0:3, 0x2a6)
    |-- HUB Interface - ISA Bridge                         (0:1:0, 0x1b2)
    |-- SMBus Controller                                   (0:1:1, 0x1b4)
    |   |-- PIC16LC System Monitor
    |   |-- Conexant CX25871 Video Encoder
    |   |-- ADM1032 System Temperature Monitor
    |   `-- 256 bytes Serial EEPROM
    |-- OHCI USB Controller                                (0:2:0, 0x1c2)
    |-- OHCI USB Controller                                (0:3:0, 0x1c2)
    |-- MCP Networking Adapter                             (0:4:0, 0x1c3)
    |-- MCP APU                                            (0:5:0, 0x1b0)
    |-- Audio Codec Interface                              (0:6:0, 0x1b1)
    |-- Simple Communication Controller - Generic Modem    (0:6:1, 0x1c1)
    |-- IDE Controller                                     (0:9:0, 0x1bc)
    |   `-- Primary IDE Channel
    |       |-- Western Digital/Seagate 8/10 GB Hard Drive
    |       `-- Thomson/Philips/Samsung DVD Drive
    |-- PCI Bridge                                         (0:8:0, 0x1b8)
    `-- AGP Host to PCI Bridge                             (0:30:0, 0x1b7)
        `-- NV2A GeForce3 Integrated GPU                   (1:0:0, 0x2a0)
```

1. ábra Az Xbox felépítése fanézetben. A jobb oldalon szereplő számok a PCI-eszközhelyeket és -azonosítókat adják meg



2. kép A KDE munkaasztal 640×480 felbontásban, egy Xboxon futó Mandrake Linux 9.0-val



3. kép Az Xbox szétszedve. A Philips DVD-meghajtó a bal oldalon, a Seagate merevlemez pedig a jobb oldalon látható. A háttérben látható zöld áramkör egy modlapka, ami most egy számítógép párhuzamos kapujára csatlakozik

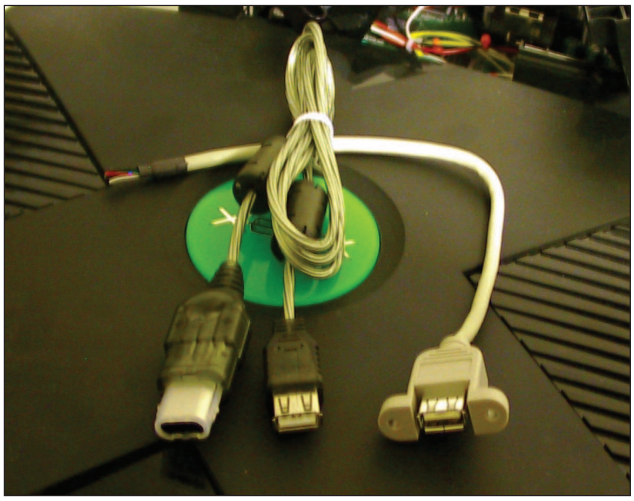
2048-bites RSA kulcsával vannak aláírva. Ha csak egyetlen bájt is módosul a fájlban, az aláírás azonnal érvénytelenné válik, a fájlt pedig az Xbox rendszermagja visszautasítja. Mivel a Microsoft titkos kulcsával nem rendelkezik, az Xbox Linux Project nem tud érvényes aláírást készíteni – vagyis nem tud olyan futtatható fájlt előállítani, amit egy gyári állapotú Xbox elfogadna. Két lehetőség kínálkozik saját kód futtatására: az egyik a ROM kicserélése, a másik egy olyan játék keresése, aminek a hibáját kihasználva futtathatjuk a kódunkat.

A legtöbben úgy veszik rá az Xboxot a Linux futtatására, hogy felnyitják a dobozt, és egy másik ROM-lapkát tesznek bele az eredeti helyett. Az úgynevezett modlapka vagy a Microsoft-féle ROM buherált változatát tartalmazza, amiben néhány egyéb apróság mellett az aláírás és a merevlemez ellenőrzése is ki van kapcsolva; vagy egy olyan „tiszta” alapprogramot, ami a normál PC-k jellemzőivel ruházza fel az Xboxot. Bár az Xbox Linuxhoz tartozik olyan rendszertöltő, aminek révén a Linux megtört Microsoft ROM-mal is futtatható (magát a ROM-ot a linuxos oldalakon nem találni meg, de az Interneten fellelhető), jogi okok miatt inkább az Xbox Linux Project Cromwell névre keresztelt tiszta ROM-ját érdemes használni. A Cromwell ROM ugyanakkor nem alkalmas az Xbox-játékok futtatására.

A gyári ROM-ot helyettesítő modlapkák számos videojátékokkal foglalkozó internetes üzletből megvásárolhatók, körülbelül 12 ezer forintos áron. Az első nemzedékbeli modlapkákat az eredeti flashlapkával párhuzamosan kötve kellett az Xbox alaplapjára forrasztani, körülbelül harminc vezetéket felhasználva. A második nemzedékbeli modlapkák az Xbox-alaplap LPC buszára csatlakoztak, és már csak kilenc vezeték felhasználását igényelték. A jelenlegi lapkák egyszerűen, forrasztás nélkül rácsavarozhatók az alaplapra. Általában üresen szállítják őket, és képesek önmaguk teljes kikapcsolására, tehát ha Clean (tiszta) Xbox Linux BIOS-t használunk, akkor az Xbox-játékok futtatásáról sem kell lemondanunk.

Mivel az eredeti ROM-tartalom az Xbox alaplapján újraprogramozható flashlapkában található, felülírásával állandó jelleggel átalakított géphez jutunk, és külön eszközök beszerelésére sincs szükség. Mindezt úgy végezhetjük el, hogy telepítünk egy modlapkát, összekötjük az alaplap két megadott pontját, letiltva ezáltal a flashlapka írásvédelmét, futtatjuk a Linuxot, letiltjuk a modlapkát, végül egy raincoat nevű alkalmazással átprogramozzuk a beépített flashmemóriát. Ezt követően a modlapkát eltávolíthatjuk, vagyis egyetlen ilyen lapkával tetszőleges számú Xboxot tehetünk Linux-futtatásra alkalmassá.

Nemrég egy névtelenségbe burkolózó személy kihasználható hibát talált az ***Electronic Arts – 007 Agent Under Fire*** című játékában. Az Xbox-fórumon megjelent hozzászólásában pontosan leírta, hogyan használható egy módosítva mentett játék a Linux rendszertöltőjének futtatására. Ezzel a módszerrel el tudott indítani egy Linuxot, ami alól az alaplap írásvédelmi pontjainak összekötése után az alaplapi flashmemóriát át tudta programozni, vagyis az átalakítást modlapka használata nélkül



4. kép USB átalakítókábelt könnyen készíthetünk az XBox vezérlőjéhez való hosszabbítókábel és a jobb oldalon látható mama USB-csatlakozó felhasználásával. Előre legyártott kábelt is vásárolhatunk, a bal oldalon egy ilyen látható

5. kép Egy német adatközpont egy Xbox alapú Domino Servert használ, ami egy IBM pSeries géppel fürtözve fut. Az Xbox a bal oldalon látható kis doboz

végezte el. Ez a legegyszerűbb és legolcsóbb módja annak, hogy egy Xboxot Linux futtatására alkalmassá tegyünk.
Az eddig ismertetett eljárások a jelenlegi időpontig megjelent Xbox konzolokra vonatkoznak. A Microsoft folyamatosan módosítja az Xbox felépítését. Mire ez az írás megjelenik, elképzelhető, hogy az Xbox alaplapján új elrendezést alakítanak ki, és eltávolítják az LPC buszt vagy megszüntetik a flashlapka átprogramozásának lehetőségét. Ezzel a témával kapcsolatban az Xbox Linux weboldalán találjuk meg a naprakész tudnivalókat.

USB

Ha van egy módosított Xbox-készülékünk, amelyben akár a megpiszkált Microsoft BIOS, akár az Xbox Linux BIOS fut, elkezdhetjük a Linux telepítését – csakhogy hogyan létesítsünk kapcsolatot a telepítővel? Az Xbox USB kapcsolatokat használ, vagyis minden vezérlő, memóriaegység és az Xbox Live fejhallgató is USB 1.1 szabvány szerinti eszköz, de módosított csatlakozókkal. Egy megfelelő átalakító segítségével tetszőleges USB-s billentyűzet, egér, webkamera, nyomtató vagy képolvasó csatlakoztatható a géphez, ha azt a Linux PC-s változata támogatja. Ilyen átalakítókat 2500–4000 forint közötti áron vásárolhatunk az Interneten, illetve némi munkával mi is elkészíthetjük őket. Mindössze az Xbox vezérlőjéhez készült kábelt kell beszerezni, amelynek az egyik végén rajta van az Xboxba illesztendő csatlakozó, továbbá egy mama USB-csatlakozó szükséges. Mindkét kábel négy különböző színű vezetéket tartalmaz. Egyszerűen csak el kell vágni a kábeleket, majd a vezetékeket összekötve máris egy USB mama–Xbox papa átalakítókábelt kapunk.
A vezérlőn található, a memóriaegység számára kialakított aljzat ugyancsak egy USB-kapu, tehát az USB mamacsatlakozót ide is forraszthatjuk.
Ha az USB-s billentyűzetet és egeret még nem vettük meg az Xboxhoz, próbáljunk olyan billentyűzetet szerezni, ami beépített USB vagy PS/2 csatlakozóval is rendelkezik az egér számára. Így egyetlen Xbox–USB-átalakítóval is boldogulhatunk. A Macintosh billentyűzetek szintén jól használhatók – és legalább nem találunk rajtuk Windows-gombokat.

Terjesztések

Megvan a módosított Xbox, sikerült billentyűzetet és egeret csatlakoztatnunk hozzá; most eldönthetjük, hogy az Xbox Linuxot magunk fordítjuk-e le, vagy az előfordított változatok valamelyikét telepítjük fel. Az Xbox Linux jelenleg három fő terjesztést bocsát az érdeklődők rendelkezésére: Mandrake, Debian és Xbox Linux Live System. Az utóbbi egy X nélküli, de a Trolltech Qtopiát tartalmazó Linux-összeállítás. Nem települ fel a merevlemezre, és az Xbox vezérlőjével kezelhető. Lényege az, hogy a kezdőknek némi rálátást ad a Linux által elérhető lehetőségekre: a Mandrake és a Debian teljes terjesztések, ezek a merevlemezre kerülnek fel. A Mandrake 9.0 jelenleg is hozzáférhető, a 9.1 nemsokára megjelenik. Mindkettő a Mandrake Linux PC-s változataira épül, és teljes mértékben képes együttműködni azokkal, illetve RPM-csomagjaikkal. Megtalálható bennük a Gnome és a KDE, illetve számos további népszerű alkalmazás. A Debian Xboxra készült változata CD-ről is indítható X-felülettel, de természetesen merevlemezre is telepíthető. A Debian esetében rövidebbek a megjelenési ciklusok, és frissítések is gyakrabban látnak napvilágot – éppen ezért vonzódik hozzá a legtöbb fejlesztő.

A rendszertöltő

Akik nem akarnak előre elkészített terjesztéseket használni, hanem minden aprósággal maguk szeretnének elbíbelődni, illetve akik meg szeretnék ismerni a rendszer működését, vessenek néhány pillantást az Xbox Linux rendszertöltőkre, valamint a rendszermag- és XFree-foltokra.
Mint már céloztam rá, az Xbox Linux az Xbox Linux Clean BIOS segítségével és aláíratlan ***.xbe***-fájlból is indítható. Az utóbbi játékprogramnak láttatja magát, és módosított Microsoft BIOS segítségével futtatja a Linuxot. Mindkét eljárás alapját ugyanaz a rendszertöltő adja, ami CD-, DVD- vagy merevlemezről betölti a rendszermagot és az indítási RAM-lemezeket, 4 MB memóriát lefoglal a videovezérlő számára, a maradék 60 MB-ot pedig a Linux-rendszermag rendelkezésére bocsátja. A rendszermag adatait, illetve az `initrd` fájlok nevét és helyét a ***linuxboot.cfg*** fájl tartalmazza, amelynek – Microsoft BIOS használatakor – ugyanabban a könyvtárban kell lennie, mint az ***.xbe*** fájlnak. A Clean BIOS használatakor a felhasználó adhatja meg a fájl helyét.

A rendszermagfoltok

Ahhoz, hogy a Linux-rendszermag Xboxon is működjön, mindössze egyetlen bájtot kell átírni –, ha azonban jól működő rendszermagot akarunk, akkor egy kicsit többet kell dolgoznunk vele. Az Xbox PCI lapkakészletének van egy súlyos hibája, ami miatt a gép összeomlik, amikor rendszerindításkor a Linux megpróbálja számba venni a PCI-eszközöket. Az összeomlás elkerülhető, ha a felsorolt PCI-eszközazonosítók tartományát szűkítjük. Egy további nehézség az óra kapcsán merül fel: az Xbox rendszerórája körülbelül hat százalékos eltéréssel ketyeg, ami miatt például a zenék lejátszása felgyorsul, a rendszermag pedig hibásan ismeri fel a processzor órajelét.
A leállítás és újraindítás folyamata is más az Xbox esetében, de ha a rendszermagot némi kóddal bővítjük, akkor képes lesz a gép megfelelő leállítására és újraindítására. Mivel más játékkonzolokon sokszor kerülték meg a másolásvédelmet úgy, hogy egy jogtiszta játékot behelyezés után azonnal egy másolt példányra cseréltek ki, az Xbox alapértelmezés szerint a lemez kiadásakor azonnal újraindul. A megfelelő programmal megelőzhetjük az újraindítást, ha a PIC16L lapka újraindításra vonatkozó kérdésére nemmel válaszolunk. A rendszermag egyik foltja erről is gondoskodik. Megint egy másik folt a DVD-meghajtót érinti, ami azt állítja magáról, hogy nem támogatja a video-DVD-ket, pedig ez nem igaz. Az Xbox Linux oldalról több elterjedt rendszermagváltozathoz tölthetünk le foltokat, ezek a fenti módosításokat mind tartalmazzák, illetve támogatják az Xbox merevlemez-felosztási módszerét és merevlemezes fájlrendszerét (FATX) is. A módosított állományok az Xbox Linux CVS révén is hozzáférhetők. A rendszermag beállításai között mindössze az Xbox támogatását kell engedélyezni.
Az Xbox ***Linux VESA framebuffer*** módban is remekül működik, ekkor a rendszertöltő egy meghatározott grafikai módot állít be, a Linux pedig átveszi ezt, és mindig a videomemóriába ír, a megjelenítő eszköz közvetlen elérése nélkül. Egy másik lehetőség a `rivafb accelerated framebuffer` illesztőprogram foltozott változatának beszerzése, ami lehetővé teszi a konzol videomódjának üzem közben történő megváltoztatását. Bármelyik megoldás mellett döntünk is, mindenképpen engedélyeznünk kell a framebuffer illesztőt a rendszermag beállításai között, mivel az Xbox Linux – jelenleg – nem támogatja a szöveges módot.
Az ALSA hangrendszer egy kisebb foltozásával – ami úgyszintén az Xbox Linux CVS révén érhető el – az Xbox hangrendszere használható az i810 illesztőprogrammal. Az nVidia nForce hálózati csatolóhoz az nVidia weboldaláról – kizárólag bináris formában – letölthető illesztőprogram bármilyen módosítás nélkül Xboxon is működik. Ha a lemezkiadással kapcsolatos javítást engedélyezni akarjuk, illetve el szeretnénk érni az alaplapi EEPROM 256 bájtját, akkor szükségünk lesz egy SMBus illesztőprogramra. Az Xbox `Linux CVS i2c-xbox` modulja és az `lm_sensors` tervezet `amd-756` modulja közül ízlés szerint választhatunk – mindkettő tökéletesen működik.

XFree86

Az XFree módosítások nélkül is azonnal futtatható, ha framebuffer illesztőprogramot használunk, illetve kikapcsoljuk a PCI-számbavételt a beállítófájlban. Az `nvdrv` illesztőprogram egy módosított változata lehetővé teszi a működés közben történő videomódváltást, illetve 2D-gyorsítást is biztosít (GLX kiterjesztések). A multimédiás alkalmazások egy meg nem jelenő gyorstárba képezhetik le ablakaikat, megjelenítésükről és megfelelő átméretezésükről már a vas gondoskodik. Az illesztőprogramból előfordított változatok is elérhetők. Az `nvdrv` nyílt forrású illesztőprogram az nVidia grafikai eszközökhöz, ami nem támogatja a 3D-gyorsítást. Már megkezdődtek azok a munkák, amelyek az nVidia részéről csak binárisan elérhető, ám 3D-támogatással rendelkező XFree-illesztőprogram megfoltozására irányulnak.

Az alkalmazások hatékonnyá varázslása

Maga az Xbox eszköz elég erős ahhoz, hogy Linux alatt DVD-t vagy DivX-filmeket nézzünk. Ha a lehető legjobb teljesítményt akarjuk elérni, akkor az alkalmazások fordításakor a tényleges eszközökhöz kell igazodnunk. A gép Celeron processzora egy Pentium III-as osztályú CPU lapka, támogatja a 686-os utasításkészletet, illetve az MMX és az SSE utasításokat. Több alkalmazás – köztük az MPlayer – ezt magától felismeri. Ha az `nvdrv` XFree-illesztőprogramot használjuk, akkor a videoalkalmazások számára engedélyezhetjük a GLX-támogatást. Például az MPlayer X alatt az `nvdrv` illesztőprogrammal a leggyorsabb, még a framebuffer-módnál is gyorsabban működik. Azt se feledjük el, hogy inkább a gép képfelbontását kell csökkenteni, és nem az alkalmazások képét kell a kimenethez méretezni. 640×480-as módban a PlayStation-emulátor `epsx` gond nélkül fut, úgy, hogy a kép 400×300 képpontra van átméretezve.
Noha az Xbox csak 64 MB memóriával rendelkezik – ezt kellő forrasztási tapasztalatok birtokában 128 MB-ra lehet bővíteni –, az asztali környezetek, mint a Gnome és a KDE, illetve az OpenOffice.org és a fontosabb alkalmazások megfelelően működnek. VMware segítségével MS-DOS és Windows 95/98/NT/2000 is futtatható Xboxon. Minimális X-környezettel, asztali kellékek és ablakkezelő nélkül a Windowsnak akár 48 MB memóriát is tudunk adni.

Összegzés

Az elterjedt PC-s Linux-terjesztések átültetett változataival, illetve a fontosabb linuxos alkalmazásokkal az Xbox készen áll arra, hogy asztali gépként, kiszolgálóként vagy multimédiás eszközként használjuk. Kiváló adottságainak köszönhetően átalakítása is több mint céltalan módosítás – értékes eszközt nyerünk vele.

Linux Journal 2003. július, 111. szám

Michael Steil (mist@c64.org)
Informatikushallgató a Mücheni Műszaki Egyetemen. 2002 májusában ő hívta életre, és máig ő irányítja az Xbox Linux Projectet. Weblapján keresztül ➲ http://www.michael-steil.de érhető el.

KAPCSOLÓDÓ CÍMEK

Az összes modlapka áttekintő oldala
➲ http://www.xbox-scene.com/modchips.php
USB átalakítókábel ➲ http://www.lik-sang.com
és ➲ www.googleshack.com
Xbox Linux-weboldal (források és bináris állományok letöltése, számos útmutató, hivatkozás stb.)
➲ http://xbox-linux.sourceforge.net
Xbox-oldalak
➲ http://www.xboxhacker.net és www.xbox-scene.com

AMD64 Opteron: az első találkozás

Az Opteron felépítésénél fogva megtartja az x86 alapú gépekkel a bináris szintű együttműködés lehetőségét, ám megnövelt címterével komolyabb alkalmazások – például áramkörtervező programok – kezelésére is alkalmas.

Még az AMD64 Opteron processzor hivatalos megjelenése előtt módunk nyílt arra, hogy kipróbáljunk egy kétprocesszoros SMP rendszert egy 64 bites Linux-terjesztéssel. A legtöbb Linux-felhasználó számára az Opteron bevezetése az elmúlt évek legjelentősebb eseménye. Előzetesünkben a következő témákkal fogunk foglalkozni:

- Az AMD64 Opteron felépítésének áttekintése.
- Kétutas SMP mintarendszer a Newisys, Inc.-től.
- Néhány futtatási eredmény GPL alkalmazásokkal.
- A teljesítményadatok előzetes értékelése.

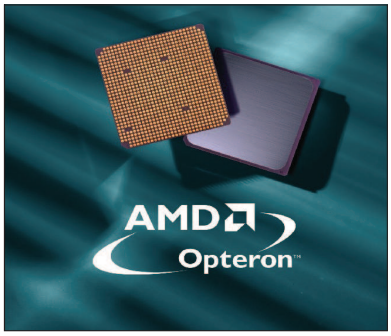
AMD64 Opteron processzorok

Az Opteron egy új eszközcsalád, ami ugyan 64 bites felépítéssel készül, ám képes együttműködni a korábbi 32 bites, x86 alapú eszközökkel. Az AMD döntése, mely szerint fenntartja az együttműködés lehetőségét, kedvezően befolyásolta a 32 bites alkalmazások áttelepítésének lehetőségeit.

Az Opteron négy programozási modellt támogat. Az első az általános célú modell, ami alapvető műveletek végrehajtására szolgál, például ilyen a memória-hozzáférés, a folyamatvezérlés, a kivételkezelés, a ki- és bevitelek. Az általános célú modell olyan memória-kezelésbeli egyszerűsítéseket is életbe léptet, aminek előnyeit a többi alkalmazáskészítési modell is kiaknázza. A második modell a 128 bites médiaprogramozás, ami 128 bites XMM regisztereket használ. Ez a modell többek között egész és lebegőpontos jellegű műveletek végrehajtását támogatja vektorokon és skaláris adatokon. A 64 bites médiaprogramozási modell hasonló képességekkel rendelkezik. Az utolsó programozási modell az x87 lebegőpontos, ami x87-es regisztereket használ 80 bites lebegőpontos és skaláris jellegű műveletek végrehajtására.

A processzor magjának kódneve Sledgehammer, magát a lapkát 940 érintkezős kerámia mikroPGA tokozásban kapjuk. A jelenlegi Opteronok közel 106 millió tranzisztort tartalmaznak, 130 nm-es csíkszélességgel, és „szilícium a szigetelőn” (SOI) eljárással készülnek. A lapkák gyártása az AMD Fab30 gyárában, Drezdában folyik. Az elsőszintű (L1) gyorsítótár mérete 128 KB, ez két részre oszlik: egy 64 KB méretű utasítás- és egy 64 KB-os adatgyorsító tárra. A lapkán egy 1 MB kapacitású másodszintű gyorsítótár is helyet kapott. A processzor 1,55 voltos feszültséggel üzemel, szilíciumlapkájának mérete 193 mm^2. A források között további adatokat is találni az AMD64 felépítéséről, illetve a processzor nyílt forrású támogatásáról.

Az Opteron processzor magas fokon integrált lapka, adottságai a kiegyensúlyozott rendszerteljesítmény elérését szolgálják. Beépített, nagyteljesítményű adatkapcsolattal bír – ennek neve HyperTransport –, ami 6,4 GB/s sebességű, teljes kétirányú adatcsere lehetőséget teremt a processzor és a többi HyperTransport-csomópont között. A processzor akár három HyperTransport-összeköttetést is tud kezelni, így összesített sávszélessége a 19,2 GB/s-t is elérheti. Emellett minden Opteron beépített memóriavezérlővel rendelkezik, ami rendkívül nagy sávszélességet és hibajavítási lehetőségeket biztosít. Az ECC (hibajavító kód) alapú védelem az első- és másodszintű gyorsítótár tartalmára, a másodszintű gyorsítótár jelölőire (tag) és – megfelelő memória telepítése esetén – a külső DRAM adataira egyaránt kiterjed.



Az AMD az Opteronok esetében háromszámjegyű jelölést alkalmaz. Az első számjegy az SMP méretezhetőséget adja meg, ami az 1,8 GHz órajelű, 244-es és e cikk alapjául szolgált, 1,6 GHz órajelű, 242-es jelzésű Opteron esetében kettő, vagyis ezek a processzorok kétutas rendszerekben használhatók. A második számjegy az adott méretezhetőségi osztályon belüli teljesítményre utal. Ahogy a lapkagyártási megoldások fejlődnek, a költségek pedig csökkennek, az osztályokon belül további modellszámok is megjelennek majd, nagyobb órajellel, alacsonyabb áron.

Az AMD egyutas Opteront is gyártani fog, amit nagy teljesítményű, de viszonylag olcsó gépekbe szán.

A 240-es, 242-es és 244-es modellek már cikkem írásakor is elérhetők. A nyolc- és egyutas modellek később jelennek meg, még ebben az idén.

Az Opteron az x86 kiterjesztése, általa a meglévő 32 bites alkalmazások 64 bites operációs rendszer felett is futtathatók. A 64 bites operációs rendszert futtató felhasználók a jövőben megjelenő, 64 bites alkalmazásokat is használni tudják majd, miközben saját kedvük szerint válthatják le korábbi 32 bites alkalmazásaikat.

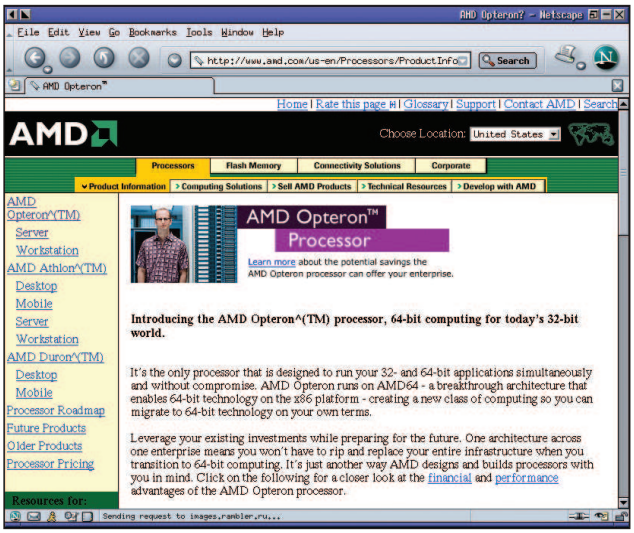
A Newisys 2100 kiszolgáló

A próbagép a 2100 jelzést kapta, gyártója a Newisys, Inc. – egy olyan cég, ami immár két éves tapasztalattal rendelkezik az Opteronnal kapcsolatban (lásd a *Kapcsolódó címeket*).

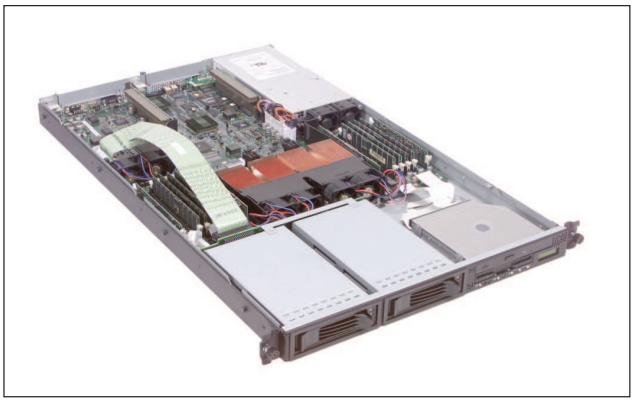
A 2100-at, ami egy 1 egység magas, állványra szerelhető rendszer, remekül összerakták. Mechanikai és elektronikus felépítése a doboz zsúfoltsága ellenére megbízhatóságot sugall (2. *kép*). A tápegység MTBF (a meghibásodások között átlagosan eltelt idő) mutatója például 500 000 óra. Ha egy évben 2080 órát dolgozunk, vagyis napi 8 órában, akkor ilyen szintű megbízhatósággal átlagosan 240 évente volna szabad hibát ejtenünk.

Az Opteronhoz igazított lapkakészleteknek és a nagyszerű adottságoknak köszönhetően a rendszer kiemelkedő memóriakezelési teljesítménnyel és -kapacitással rendelkezik. Eredményként egy nagyteljesítményű, kiegyensúlyozott kiszolgáló jött létre, erőteljes be- és kiviteli alrendszerrel. A próbagép 6 GB PC2700-as memóriát tartalmazott, bár akár 16 GB kezelésére is képes.

A *3. kép* felülnézetben mutatja a rendszert. A két réz hűtőborda alatt lapulnak a processzorok. A gép kétféle sebességet támogat: az 1,6 GHz-es 242-es és az 1,8 GHz-es 244-es Opteron-modelleket. Az Opteronok HyperTransport alapú összeköttetést létesítenek egymással és a lapkakészlettel. A processzor–processzor kapcsolatátviteli sebessége mindkét irányba 3,2 GB/másodperc. Az Opteronok mindegyike saját, belső memóriavezérlővel bír, ami támogatja az ECC DDR SDRAM-okat, és – mindegyik processzorhoz külön-külön – legfeljebb 5,33 GB/másodperc sávszélességre képes. A gépben két memóriabankot találunk, mindkét processzorhoz egyet-egyet.



1. kép Az AMD honlapja



2. kép A Newisys 2100 két Opteron processzort tartalmaz egy 1 egység magas házban

Az AMD-8000 HyperTransport lapkakészlet egy AMD-8131 HyperTransport PCI-X lapkát és egy AMD-8111 I/O jelzésű elosztólapkát foglal magában. Az AMD-8131 lapka a teljes hosszúságú PCI-X kártyákat 64 biten és 133 MHz órajellel, 1 GB/másodperc átviteli sebességet elérve hajtja, míg a félhosszúságú PCI-X kártyákat 64 biten és 66 MHz órajellel, 0,51 GB/másodperc sebességet biztosítva. Az AMD-8131 két darab 10 Mbit/100 Mbit/1 Gbit másodpercenkénti sebességeket támogató hálózati csatolót bocsát rendelkezésünkre, illetve egy kettős Ultra-SCSI RAID-vezérlőt. A próbagépben üzem közben cserélhető, Ultra-SCSI meghajtókat találtunk RAID-összeállításban, a gépház elülső részében. Az AMD-8111 lapka VGA-, IDE CD-ROM- és USB-kapuval is rendelkezik. Egy SuperI/O lapka tőle függetlenül biztosítja a hajlékonylemezes meghajtó, a billentyűzet és az egér csatlakoztatásának lehetőségét, illetve egy hagyományos soros kaput is igénybe vehetünk.

A rendszer egy teljesen különálló beágyazott felügyeleti processzorral rendelkezik, ami Linuxot futtat. Ez az alrendszer a Motorola XPC855T PowerPC processzorára épül, és 2.4.18-as rendszermagot futtat. Az apró előoldali felügyeleti konzol csatlakozó mellett a felügyeleti processzor két saját, 10/100 Mb/s sebességű ethernetcsatolóval is rendelkezik, amik független felügyeleti alhálózatra köthetők. A rendszerfelügyeleti teendők tehát billentyűzet és monitor nélkül végezhetők el, még soros konzolos hozzáférés-kiszolgálóra sincs szükség, és a PCI-foglalatok közül sem kell felhasználni egyet sem.

A rendszerfelügyeleti képességek tekintetében nincs okunk panaszra. A felügyeleti processzor az SNMP, a CIM és az IPMI protokollt támogatja. Ugyancsak ismeri a NIS, a Microsoft Active Directory és az LDAP alapú hitelesítést. A felügyeleti processzor beállításait pont–pont-kapcsolaton keresztül tudjuk másolni. Emellett arra is lehetőség van, hogy egy adott felügyeleti processzort egy egész kiszolgálótelep vezérlőjeként jelöljünk ki.

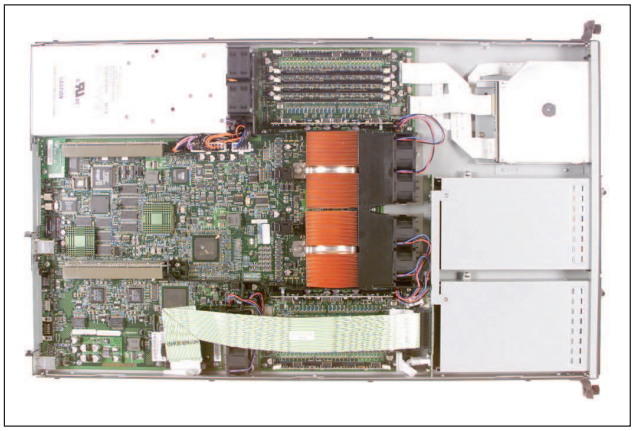
A felügyeleti processzor akár teljesen szűz gép ellenőrzésére is használható. A gép vizsgálata – a memória és a processzor esetleges hibáinak felderítése – teljesen önállóan is elvégezhető.

A Newisys bejelentette, hogy ők a műszaki megoldásokat fogják fejleszteni, de a gyártást és értékesítést külső OEM és szerződéses gyártók számára fogják átadni. A Sanmina SCI-vel létező szerződésük van a gyártásra, míg az értékesítést az Avnet végzi. A Newisys partnerei között nagyobb informatikai cégeket is találunk, mint az Angstrom Microsystems, az APPRO, a RackSaver, az M&A Technology, a Microway, a New Technology Solutions, Inc. és a ProMicros. Írásom készültekor körülbelül hatszáz próbagép már az OEM-eknél, Fortune 500 cégeknél és természetesen a Linux Journal szerkesztőségében vendégeskedett.

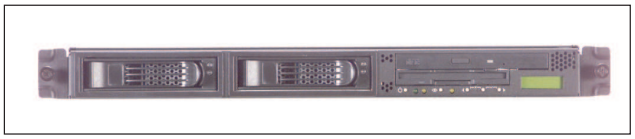
SuSE Linux Enterprise Server 8 és GNU-alkalmazások

Mivel nagyon új gépről van szó, az `uname -a` parancs után a `cat /proc/cpuinfo` szöveget gépeltük be izgatottan a bash parancssorába:

```
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 15
model           : 5
model name      : AMD Opteron(tm)
stepping        : 0
cpu MHz         : 1594.286
cache size      : 1024 KB
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr pae
                  mce cx8 apic sep mtrr pge
                  mca cmov pat pse36 clflush
                  mmx fxsr sse sse2 syscall
                  nx mmxext lm 3dnowext 3dnow
bogomips        : 3178.49
TLB size        : 1088 4K pages
clflush size    : 64
address sizes   : 40 bits physical, 48 bits
                  virtual
power management : ts ttp
```



3. kép A Newisys 2100 felülnézetből



4. kép A kiszolgáló felügyeleti processzora saját LCD-vel bír, ami a CD-ROM-meghajtó alatt található

Az 1-es processzor adatai is ugyanezek, ám annál a BogoMips-érték 3185,04.
A SuSE Linux Enterprise Server 8 elsődleges grafikus felülete a KDE 3.0. A gépen azokat a megszokott segédprogramokat és alkalmazásokat találtuk, amiket a SuSE-terjesztésekben általában szoktunk. Az, hogy a SuSE Enterprise Linux immár Opteron processzoron is fut, rengeteg munka eredménye, és a 64 bites Linux megteremtésének régre visszanyúló történetét idézi fel – az egész valamikor akkor kezdődött, amikor ***Linus Torvalds*** egy Digital Alpha gépet kapott ***Jon „maddog" Hall***-tól. A GNU-környezetről egyszerűen csak annyit, hogy működik – 64 biten. A megszokott fordítókat és a GNU alkalmazásfejlesztési környezetet minden módosítás nélkül azonnal használatba vehetjük. Az első meglepő dolog, hogy az Emacs gyakorlatilag azonnal elindul, még pislogni sincs időnk.
Első próbaként lefordítjuk és futtatjuk a GNU Scientific Library (GSL) könyvtárat, ami numerikus számítások elvégzésére szolgál. A könyvtár elég erőteljes, mi az 1.3-as változatát használtuk. Ezután lefordítottunk néhány példaprogramot a GNU Scientific Library Reference Manualból, és véletlen számokat állítottunk elő. Maga a program meglehetősen egyszerű, fordításakor semmilyen optimalizálást nem végeztünk, ám a gép még így is háromszor gyorsabbnak bizonyult egy 800 MHz-es x86-os gépnél. Hamarosan azt is megtudtuk, a hétköznapok során mennyire nem foglalkozunk azzal a ténnyel, hogy a gép, amin programozunk, 32 bites – vagy éppen nem az. A tesztprogram egyik része egy 1-es értékű egész számot a `sizeof(int) * 8 - 1` értékkel balra tolt el, a gép natív egész számainak bitmérete alapján így kapta meg a bitek számát. Ezen a gépen – ellentétben a normál x86-os gépekkel – a programot ezzel a hordozhatónak éppen nem nevezhető paranccsal lefordítva azonnal egésztúlcsordulást kaptunk. Ugyan a fontosabb programok 64 biten is mind tökéletesen futnak, előfordulhat, hogy a saját kódunkkal dolgoznunk kell egy kicsit.
A következő próbát az Icarus Verilog fordítóval végeztük. Az Icarus Verilog egy GPL alatt megjelenő Verilog HDL fordító elektronikai tervezési feladatok automatizálására (EDA), különös tekintettel a HDL-szimulációkra és összeállítások tervezésére. (A Linux Journal korábban kétszer is készített interjút ***Stephen Williams***-szel, az Icarus készítőjével.) A fordító szolgáltatásainak lendületesen bővülő körét szemlélve megállapítottuk, hogy komoly, logikai és FPGA tervezésre ipari környezetben is alkalmas EDA eszközzé nőtte ki magát. Mivel Stephen az elmúlt öt évben az Icarust tisztán 64 bites kóddal készítette, Alpha gépen – és Linux alatt – a fordító lefordítása nem okozott gondot. Érdekes volt nézni, hogy milyen gyorsan futott és milyen jól bírta a nagyobb terheléseket. A próbát egy nagyobb szorzólogika modelljével és az Icarus próbakörnyezetével végeztük, és ugyanazt a binárist használva közel kétszer gyorsabb futás értünk el, mint egy 1,5 GHz-es órajelű Athlon processzorral. A nagyobb terhelést úgy szimuláltuk, hogy egy 1 720 648 sornyi Verilog-kódból álló logikai modellt dolgoztunk fel. Az eredmény lélegzetelállító volt. A gép 61 perc alatt fordított le egy olyan modellt, aminek a memóriabeli lenyomata nagyobb volt, mint a 32 bites Linux alatt elérhető legnagyobb felhasználói memóriaterület – nem kevesebb, mint 3,6 GB.
Mivel az Icarus a saját szálkezelő rendszerét használja, ezek az EDA-próbák csak a gép felét használták ki. A másik processzor – méretezhető memória-sávszélességgel – további EDA-szimulációkat tud futtatni úgy, hogy ez az első processzor munkáját nem érinti. Az Opteron egyértelműen kiváló választás, ha mérnöki tervezőmunkákról van szó.
Az utolsó nagyobb próba a Linux Test Project (LTP) lefordítása

1. táblázat Az Opteron 244 teljesítményadatai

Név	Alap	Csúcs
SPECint_rate2000	24,5	26,1
SPECfp_rate2000	24,6	26,5

2. táblázat Az Opteron 244 Java Middleware Benchmark eredményei

Név	Művelet/másodperc
SPECjbb	55 000

KAPCSOLÓDÓ CÍMEK

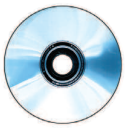
Webes források
AMD64 Opteron, Advanced Micro Devices
➲ http://www.amd.com
Newisys, Inc. (műszaki fejlesztés) ➲ http://www.newisys.com
Az AMD64 termékvonal nyílt forrású támogatása
➲ http://x86-64.org

GNU-programok
GNU Scientific Library
➲ http://www.gnu.org/directory/GNU/GNUsl.html
Icarus Verilog fordító ➲ http://www.icarus.com
Linux Test Project ➲ http://ltp.sourceforge.net

Teljesítménytesztek
HPC Linpack ➲ http://www.netlib.org/linpack
SPEC CPU2000 ➲ http://www.spec.org/osg/cpu2000
SPEC JBB2000 ➲ http://www.spec.org/jbb2000
SPECWeb 99 ➲ http://www.spec.org/osg/web99
SPECWeb SSL ➲ http://www.spec.org/osg/web99ssl
TPCC ➲ http://www.tpc.org/tpcc

SQL adatbázis beágyazása az SQLite segítségével

Ha élvezni szeretnéd az SQL kényelmét, de nem akarsz egy nagyméretű adatbázis-kiszolgáló telepítésével szenvedni, ágyazd be az SQLite-ot a programodba, bármilyen nyelven íródjon is.



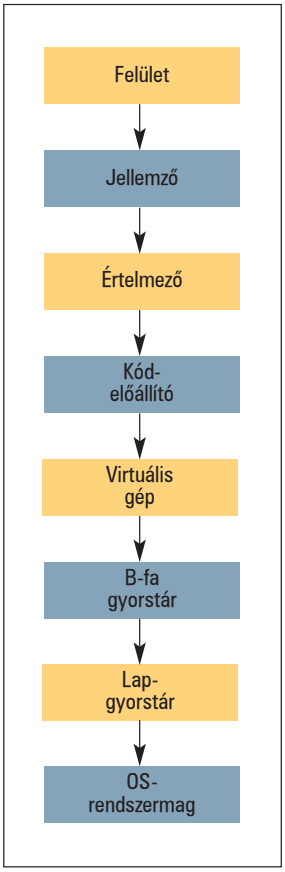
Az SQLite nagy teljesítményű, beágyazott relációs adatbázis-kezelő rendszer, ami egy viszonylag kisméretű C-könyvtárban kapott helyet. Fejlesztője ***D. Richard Hipp***. Támogatása az SQL92 szabvány túlnyomó részére, több tábla és index létrehozatalára, tranzakciókra, nézetekre, ravaszokra és rengeteg különböző ügyfélfelületre és illesztőprogramra terjed ki. A könyvtár önálló egységet képez, és mindössze 25 000 sornyi ANSI C-ben készült kódból áll. Bármely célra szabadon felhasználható; gyors, hatékony és méretezhető, számos különböző rendszeren és géptípuson fut, az ARM/Linux párosítástól kezdve egészen a SPARC/Solarisig. Adatbázis-formátuma bájtszinten alkalmas a különböző bájtsorrendet használó gépek közötti együttműködésre, adatbázisainak mérete pedig akár 2 TB is lehet.

Hipp agyából akkor pattant ki az SQLite ötlete, amikor a General Dynamicsnél dolgozva csapata az amerikai haditengerészet DDG osztályú rombolói számára fejlesztett fedélzeti programot. A program HP-UX alatt futott, és Informix adatbázist használt. Miután elkezdték a munkát, hamarosan felismerték, hogy az Informixszel nagyon nehéz dolgozni. Ha egyszer sikerült üzembe helyezni, akkor elég szépen futott, ám még egy nagy tapasztalattal rendelkező szakembernek is egy teljes napja ráment egy-egy telepítésre vagy frissítésre. Abban az időben a csapat a fejlesztésekhez Linuxot és PostgreSQL-t használt. A PostgreSQL lényegesen kevesebb felügyeletet kívánt, ám ők olyan önálló programot akartak készíteni, ami bárhol futtatható, függetlenül az adott rendszerre telepített egyéb programoktól. 2000 januárjában Hipp és egy munkatársa megvitatták egy egyszerű, felügyeletet és támogatást nem igénylő, beágyazott SQL-adatbázismotor készítésének az ötletét, aminek alapja GDBM lenne. Később Hipp egymaga látott neki a munkának, és hamarosan elkészült az SQLite 1.0.

A General Dynamics a PostgreSQL helyett azonnal az SQLite-ot kezdte alkalmazni. Az SQLite segítségével önálló alkalmazásokat tudtak készíteni, amiket gyorsan és könnyen telepíthettek a bemutatókra és vásárokra vitt hordható (wearable computer) és hordozható számítógépekre. Az Informix továbbra is fontos szerepet játszik a fedélzeti rendszerekben – igaz, Hippet nemrég felkérte a tengerészet, hogy segítsen az SQLite HP 9000-re való lefordításában. Lehet, hogy hamarosan változások lesznek?

A nagyobb módosítások a 2.0-s változattal érkeztek. Az első változat az adatok tárolásához GDBM-et használt, ami rendezetlen kulcsokat, másszóval kivonatolást alkalmaz. Emiatt azonban korlátozott volt az SQLite szolgáltatásainak a köre. Emellett a GDBM GPL szerződéssel jelenik meg, és ez néhányakat visszatart a kipróbálásától. 2001 januárjában Hipp egy saját B-fa alapú háttéren kezdett dolgozni, amivel a GDBM-et akarta kiváltani. Az új B-fa alrendszer kulcsok szerint rendezve tárolja a rekordokat, így hatékonyabb működést, logaritmikus időben végrehajtott minimum- és maximumkeresést, egyenlőtlenségi megszorításokkal végrehajtott indexelt lekérdezéseket tesz lehetővé, illetve a tranzakciókat is támogatja. A végeredmény egy sokkal nagyobb tudású adatbázisrendszer lett. A 2.0-s változat 2001 szeptemberében jelent meg public domain szerződéssel.

Az SQLite felemelkedése is a 2.0-s változat megjelenésével indult meg. Ekkor már egyre több, különféle területeken dolgozó fejlesztő ragadott billentyűzetet, és írta le, hogyan építette be kereskedelmi vagy éppen szabad termékekbe. Néhányan műszaki támogatást vagy egyedi módosításokat is kértek Hipptől. Hipp elmondása szerint az egyik széles körben elterjedt windowsos program újabb kiadásai is magukban foglalják az SQLite egy módosított változatát. Az SQLite nemcsak a fejlesztést eredetileg ösztönző haditengerészetnél, de az amerikai hadsereg egyéb részlegeinél és a Duke Energynél is megtalálta helyét. Másfél évvel ezelőtti nyilvános megjelenése óta az SQLite szolgáltatásainak és felhasználóinak köre rohamosan bővül. Az SQLite Wikibe bepillantva számos további alkalmazást találunk, ezeknek a fejlesztői felfedezték és a saját programjukba építették be az SQLite-ot. Saját Apache modullal is rendelkezik (`mod_auth_sqlite`), aminek létezését már önmagában is a siker jelének tekinthetjük. ***Gerhard Häring*** és jómagam a PySQLite – Python kiterjesztés SQLite-hoz – készítőiként meglepődve számláltunk össze több mint 3000 letöltést, kevesebb mint egy év alatt. Az SQLite jelenleg a legmagasabb pontszámot elért adatbázismotor a ➲ http://freshmeat.net oldalon.



Az SQLite felépítése

Felépítés

Az SQLite tetszetős, moduláris felépítésű. Nyolc elsődleges alrendszerre osztható (lásd az *ábrán*), ezek közül néhány meglehetősen érdekes megközelítést tükröz az adatbázis-kezelés területét illetően.

Az ábra tetején az értelmező és a jelkezelő látható. Az SQLite saját, magas fokon hatékonnyá tett és a Lemon névre keresztelt

1. lista Az explain kimenete egy egyszerű lekérdezésre

```
SQLite version 2.7.1
Enter ".help" for instructions
sqlite> .explain
sqlite> explain select lastname,firstname
        from person;
addr   opcode         p1     p2    p3
----   -------------  ----   ----  ---------
0      ColumnCount    2      0
1      ColumnName     0      0     lastname
2      ColumnName     1      0     firstname
3      Open           0      3     person
4      VerifyCookie   276    0
5      Rewind         0      10
6      Column         0      4
7      Column         0      3
8      Callback       2      0
9      Next           0      6
10     Close          0      0
```

értelmezőt tartalmazza, ami gyors, hatékony kódot készít, és újszerű felépítésénél fogva kifejezetten ellenáll a memóriaszivárgásoknak. Alul a Knuth alapú B-fa megvalósítás található, ami egy hangolható lapgyorsítótár felett fut, a lehető legkisebb számú lemezhozzáférés mellett. A lapgyorsítótár egy operációs rendszertől függő elvonatkoztatási rétegen fut, ennek köszönhetően a könyvtár magasabb fokon hordozható.

A könyvtár középpontjában a virtuális adatbázismotor (VDBE) található. A VDBE végzi az összes adatkezeléssel kapcsolatos feladatot, minden az ügyfelek és a tárrendszer között áramló adat rajta halad keresztül. Tulajdonképpen ez az SQLite szíve. A VDBE szerepe az SQL-utasítások értelmezése után kezdődik. A kódkészítő az értelmezési fa alapján egy apró programot állít elő, ami a VDBE virtuális gép nyelvében található utasítások sorozata. A VDBE egyenként végrehajtja az utasításokat, végeredményként az SQL-utasításban megadott parancsot teljesíti. A VDBE gépi nyelve 128 utasításból áll, ezek mindegyike az adatbázis-kezeléssel kapcsolatos. Külön parancsok szolgálnak a táblák megnyitására, az indexekben való keresésre, a rekordok tárolására és törlésére, illetve a számos egyéb adatbázis-kezelő műveletre. A VDBE minden utasítása egy műveleti kódból és legfeljebb három műveleti értékből áll. Egyes utasítások – jellegüktől függően – akár mindhárom műveleti értéket használják, míg mások egyiket sem. Az `open` utasítás például, ami egy tábla egy mutatóját nyitja meg, mindhárom műveleti értéket használja. Az első műveleti érték (P1) a mutató későbbi kezelésére szolgáló azonosítót tartalmazza. A második (P2) a tábla gyökér (vagyis első) lapjának helyére mutat, a harmadik pedig a tábla neve. A `rollback` utasítás – ellenpéldaként – egyetlen műveleti értéket sem igényel, a VDBE-nek csak annyit kell tudnia, hogy végrehajtsa-e a visszavonást vagy sem.

Ha érdekel, hogy az adott SQL-utasításhoz milyen VDBE-program tartozik, az SQLite parancssorának `explain` parancsával megtekintheted. Az *1. lista* erre mutat példát.

Az `explain` nem csupán akkor hasznos, ha betekintést szeretnénk nyerni a VDBE működésébe, de olyan gyakorlati teendőknél is, mint például a lekérdezések hatékonyságának a javítása. A VDBE valóban érdemes arra, hogy önmagában is kiterjedt vizsgálatok tárgyát képezze. Szerencsére ehhez minden feltétel adott: a motorhoz remek leírás tartozik, működésének elméletéről és programnyelvének műveleteiről részletesen olvashatunk az SQLite weboldalán.

A fizikai tárolás szempontjai miatt minden adatbázis egyetlen fájlban található. Minden olyan objektum tehát, ami szerepel az adott adatbázisban (nézetek, ravaszok, indexek, táblák, sémák stb.), egyetlen, az adott SQLite-adatbázist meghatározó állományban foglal helyet. Az adatbázisfájlok egységesen formázott lapokból állnak össze. A lapméret meghatározása az adatbázis létrehozásakor történik, értéke 512 bájt és 4 Gb között lehet. Alapértelmezett esetben az SQLite 1 Kb méretű lapokat hoz létre, így lehet a legjobb általános teljesítményt elérni.

A tranzakciókat egy második állomány, a napló segítségével valósítja meg, ami ténylegesen csak akkor létezik, ha egy vagy több működő kapcsolat áll fenn az adatbázissal. Minden adatbázishoz egyetlen naplófájl tartozik, ez azokat az eredeti – módosítás előtti – lapokat tartalmazza, amik tartalma a tranzakció feldolgozása közben módosult. Amikor jóváhagyod a tranzakciót, a naplólapokra többé nincs szükség, így törlésre kerülnek. A műveletek visszavonásakor a motor a naplófájlból állítja vissza a lapokat az adatbázisfájlba. A naplófájl használata teszi lehetővé azt, hogy az adatbázis akár egy összeomlást is túlélhessen, és egységes állapotba lehessen visszaállítani. Összeomlás után az első ügyfélnek az adatbázishoz történő csatlakozása egy ravasszal elindítja az előző tranzakció visszavonását. Kicsit pontosabban: amikor az ügyfél csatlakozik, az SQLite megpróbál egy új naplófájlt létrehozni, és ekkor észleli, hogy már létezik egy ilyen. Ha erre kerül sor, akkor a motor feltételezi, hogy összeomlás történt, és a régi naplófájl tartalmát visszamásolja az adatbázisba, gyakorlatilag az eredeti, összeomlás előtti állapotába állítva vissza azt. Az ügyfél csak ezt követően kapja meg a lehetőséget a munka megkezdésére.

Egyszerű API, sok nyelv

Az SQLite egy rendkívül könnyen használható API-val rendelkezik, ami mindössze három függvény révén alkalmas az SQL-utasítások végrehajtására és az adatok kinyerésére. Bővíthető, a programozó C-visszahívók formájában egyedi függvényeket és összesítéseket adhat meg. A C API a parancsfájl-készítő felületek alapja is egyben, ezek egyikét – a Tcl-felületet – a terjesztésben is megtaláljuk. A nyílt forrású programok közössége számos további ügyfélfelületet, csatolót és illesztőprogramot is kifejlesztett, ezek révén az SQLite más nyelvekkel és könyvtárakkal is használható.

A C API használatához mindössze három lépést kell végrehajtani. Az `sqlite_open()` hívással – a fájlnév és az elérési mód megadásával – lehet csatlakozni a kívánt adatbázishoz. Ezután egy visszahívó függvényre van szükség, amit az SQLite az adatbázisból lekért rekordok mindegyikére meghív. A záró lépés az `sqlite_exec()` meghívása, aminek egy karakterláncban a végrehajtandó SQL-utasítást, illetve a visszahívó függvényünkre hivatkozó mutatót adjuk át. A hibakezeléstől eltekintve végeztünk is. A 2. listában (50. CD Magazin/SQL könyvtár) erre találunk egy egyszerű példát.

Szembeötlő – és fontos – eltérés a többi adatbázis-ügyfélkönyvtárhoz képest a visszahívó függvény használata. Az egyéb ügyfél API-k használatakor meg kell várnunk az eredmény összeállítását, az SQLite esetében viszont az eredményt összeállító folyamat kellős közepébe csöppenünk, és végigkövethetjük az eseményeket. Így az adatok kinyerésében is sokkal aktívabb szerepet játszhatunk, és közvetlen hatást gyakorolhatunk a lekérdezési folyamatra. Az adatok összesítését már kinyerés közben elvégezhetjük, de akár meg is szakíthatjuk a rekordok lekérdezését. Vegyük észre, hogy az adatbázis beágyazott

3. lista Python-példa

```
import sqlite

conn = sqlite.connect(db="db", mode=077)
cursor = conn.cursor()

SQL = "select * from person order by
      ➥lastname"
cursor.execute(SQL)

row = cursor.fetchone()
while row != None:
    print "%14s, %15s" % (row[´firstname´],
                         ➥row[´lastname´])

    row = cursor.fetchone()
```

4. lista Perl-példa

```
use DBI;
my $dbh = DBI->connect
          ➥("dbi:SQLite:dbname=dbfile",
                        "","");

my $cursor;
my @rec;

my $SQL = "select * from person order by
           ➥lastname";

$cursor = $dbh->prepare($SQL);
$cursor->execute();

while(@rec = $cursor->fetchrow_array)
{
    print "$rec[0], $rec[1]\n";
}

$cursor->finish;
$dbh->disconnect;
```

jellegénél fogva alkalmazásunk legalább annyira kiszolgáló, mint ügyfél, és a visszahívó felület révén az SQLite maradéktalanul ki is használja ennek az adottságnak az előnyeit.
A normál C API mellett egy kiterjesztett API is rendelkezésünkre áll, amivel még könnyebb a rekordok lekérdezése. Az `sqlite_get_table()` használatakor visszahívó függvényre nincs szükség. Ez a függvény inkább a hagyományos ügyfélkönyvtárakra emlékeztethet, amiknél átadjuk az SQL-utasítást, majd megkapjuk az eredmény sorait. A kiterjesztett API saját függvények és összesítők hozzáadásával többek között az SQL kibővítését is lehetővé teszi – ezekről a lehetőségekről később még lesz szó.
Végül, ha valamilyen oknál fogva ODBC-felületre van szükséged, örömmel újságolhatom, hogy ***Christian Werner*** munkájának hála, erről sem kell lemondanod. Az ODBC illesztőprogram a ➲ http://www.ch-werner.de/sqliteodbc címről tölthető le.

5. lista Héjpélda

```
#Az SQL-utasítás létrehozása, elhelyezése
#a $sql változóban
read -d ´EOF´ sql<<EOF
select * from person order by lastname
EOF

# Lekérdezés
ret=$(sqlite $db "$sql" 2>&1)
if [ $? = "1" ]
then
    # Hiba történt. Hibajelzés és kilépés.
    printf "Error: $ret.\n"
    printf "SQL: $sql\n"
    exit 1
    else
        # Hiba nem történt, az eredmény
        # kiírása
        for row in $ret
        do
            IFS="|"

            # Változók értékadása
            # a rekordokból
            set $row

            # Hozzárendelések kényelmi
            # okokból
            firstname=$1
            lastname=$2
            dir=$3
            printf "%-14s %-15s\n"
                   ➥$firstname $lastname

            unset IFS
        done
fi
```

Önműködő sorszámozás

Az SQLite a sorozatokat kifejezetten nem támogatja, ám kezeli az önműködően növelt kulcsokat, a MySQL `mysql_insert_id()` függvényével egyenértékű módon. Az elsődleges kulcsot önműködően úgy növelhetjük, hogy `INTEGER PRIMARY KEY` típussal adjuk meg. Az utoljára beillesztett rekord e mezőjének az értékét az `sqlite_last_insert_rowid()` hívással kaphatjuk meg.

BLOB-ok

Az SQLite oszlopaiban bináris adatokat is tárolhatunk, azzal a megkötéssel, hogy rögzítésük csak az első `NULL` karakterrel bezárólag történik meg. Ha bináris adatokat szeretnénk tárolni, először a kódolásukat kell elvégeznünk. Az egyik lehetőség az URL-stílusú kódolás, a másik a base64. Ha nincs semmilyen különleges igényünk, akkor az SQLite két ilyen célú függvényével – `sqlite_encode_binary()` és `sqlite_decode_binary()` – ezt is könnyedén letudhatjuk.

A programszálak biztonsága

Az SQLite pontosan annyira biztonságos, amennyire maga a beágyazóprogram is az. A kérdés többé-kevésbé az `sqlite_open()` által visszaadott SQLite-kapcsolatkezelőhöz

(connection handle) kötődik. Ez az, amit nem szabad megosztani a futtatási környezetek között, minden szálnak saját példányt kell biztosítani. Ha ragaszkodunk ahhoz, hogy a szálak osztozzanak rajta, egy `mutex` segítségével kell védeni. A kapcsolatkezelőket a Unix `fork()` hívások között sem szabad megosztani. Ezt – gondolom – nem kell különösebben magyarázni. Alapelv tehát, hogy minden szál vagy folyamat saját kapcsolatkezelővel rendelkezzen, és semmi gond nem lesz.
Az SQLite a futásidejű viselkedés szabályozására pragmákat használ. A pragmák olyan beállítások, amiket SQL jellegű írásmóddal adhatunk meg. Vannak olyan pragmák, amik – például a gyorstár méretének megadása vagy a szinkrn írások engedélyezése vagy letiltása révén – a teljesítmény finomhangolására szolgálnak. Mások hibakeresésre, az értelmező és a VDBE nyomkövetésére használhatók, illetve az ügyféloldali visszahívó függvénynek átadott adatok mennyisége is hasonló módon szabályozható. Bizonyos pragmák esetében a hatás tartóssága is szabályozható, így az adott beállítás vonatkozhat csak az adott munkamenetre, de lehet állandó jellegű is.
Ha az adott oszlop típusa `BLOB`, `CHAR`, `CLOB` vagy `TEXT`, akkor – és csak akkor – az SQLite lexigrafikusan rendezi azt. Egyéb típusoknál a rendezés numerikusan történik. Az SQLite korábban kizárólag az értékük alapján rendezte az oszlopokat. Ha „ránézésre" számot talált, akkor numerikus, egyébként pedig lexigrafikus rendezést végzett. A levelezési listán rengeteg vita volt ezzel a témával kapcsolatban, így alakultak ki a jelenlegi szabályok, amik a sémában megadott típus révén teszik lehetővé az összehasonlítás módjának kiválasztását.

Parancsfájlkészítő felületek

Mint korábban említettem, az SQLite-hoz számos ügyfélfelület létezik. Ízelítőként a 2. kódrészlet C-példájának Python-változata a *3.*, Perl-egyenértékese pedig a *4. listában* tekinthető meg. Kell ennél egyszerűbb? Az SQLite parancssorból is használható, így rendszerfelügyeleti célokra is alkalmas. A már ismert példa parancssori változatát mutatja be az *5. lista*.
Végül, mivel nem vagyok Java-, Tcl-, Ruby-, Delphi-, Lua-, Objective C-, PHP-, Visual Basic-, .NET-, Mono-, DBExpress-, wxWindows-, Euphoria- vagy REXX-programozó, mindenkinek javaslom, hogy az SQLite Wiki segítségével kutassa fel a hozzá hasonló érdeklődésűeket, és használja bátran az általa választott felületet (➲ http://cvs.hwaci.com:2080/sqlite/wiki?p=SqliteWrappers).

Az SQLite kiterjesztése

Az SQLite nagyszerű C-keretrendszerrel bír, amiben saját, SQL-ből meghívható függvények és összesítők készíthetők. Egyes burkolók – mint például a Python-burkoló – azt is lehetővé teszik, hogy ezeket a bővítmény nyelvén készítsük el. Egy SQL-utasítás – például `INSERT INTO orders purchase_date values CURRENT_TIME()` – jó szemléltető alanya egy visszahívó függvénynek (lásd a 6. listát, 50. CD Magazin/SQL könyvtár). Mindössze be kell jegyezni a függvényt, és máris lehet használni (lásd a 7. listát, 50. CD Magazin/SQL könyvtár).
Az SQLite összes beépített függvénye – mint az `avg()`, `min()`, `max()` és `sum()` – ezzel az API-val van megvalósítva, az egyetlen kivétel ez alól a `typeof()`. Felhasználói összesítők ugyanilyen könnyen adhatók hozzá. A `SELECT variance(age) from population` végrehajtása például nagyon közel áll a függvények készítéséhez. Ezt azonban már inkább a kedves olvasóra bízom, amolyan házi feladatként. Egy jó tanács: a ***func.c*** fájlban érdekes példákat lehet találni. Az SQLite a függvényekhez hasonlóan az összesítőket is az API segítségével valósítja meg.

Felügyelet

Felügyeleti célokra az SQLite esetében egy `sqlite` nevű segédprogram szolgál, ami a MySQL- és PostgreSQL-használók számára nem fog meglepetéseket okozni. Héj és parancssori módja egyaránt létezik. Héjból az adott adatbázis tábláinak nevét, sémáját és indexeit tekinthetjük meg, illetve SQL-utasításokat hajthatunk végre, akár a parancssorba gépelve, akár külső állományból véve őket. Az adatok és a VDBE kimenetének megjelenítésére is jól használható.
Az adatok betöltése és kimentése héjból és parancssorból egyaránt megoldható – igaz, az utóbbival könnyebb dolgunk lesz. Ha egy fájl érvényes DDL/DML utasításokat tartalmaz (a neve legyen ***dump.sql***), akkor a következő módon tölthető be az adatbázisba (ennek neve legyen db):

```
sqlite db < dump.sql
```

Ilyenkor létrejön egy db nevű adatbázis, mégha korábban nem is létezett. Ha az adatbázist ki szeretnénk íratni, a következő módon kell eljárnunk:

```
sqlite db .dump > dump.sql
```

Az SQLite kiváló teljesítményt nyújt. Széles körű alkalmazhatósága, könnyű használata, hordozhatósága, gyorsasága, méretezhetősége, kis mérete és letisztult kódja olyan könyvtárrá teszik, ami egyetlen programozható eszköztárából sem hiányozhat. Felhasználói szerződése mindenkinek a lehető legnagyobb szabadságot kínálja. Az SQLite Project mindig örömmel fogadja az új felhasználókat és fejlesztőket, ahogy az újabb ötleteket és vitatémákat is. Remélem, hogy mindenki ugyanolyan örömmel ismerkedik meg vele és kezdi használni, ahogy én is tettem.

A cikkhez tartozó listák megtalálhatóak az 50. CD Magazin/SQL könyvtárában.

Linux Journal 2003. június, 110. szám

Michael Owens (mike@mikesclutter.com)
Programozó, eredetileg kémikus. Egy ingatlankezelő cégnél dolgozik a texasi Dallasban, ahol Linux alatt fejleszti a cég belső alkalmazásait. A PySQLite életre hívója és társfejlesztője.

KAPCSOLÓDÓ CÍMEK

Az SQLite készítője, *D. Richard Hipp* magas szintű támogatást és tanácsadást nyújt az SQLite-hoz.
A ➲ http://www.hwaci.com/sw/sqlite/support.html címen további tájékoztatás található róla.
Az SQLite weboldala felbecsülhetetlen értékű forrás, itt találjuk az SQLite SQL C API-jának teljes leírását, illetve az alrendszerek, köztük a VDBE ismertetőjét.
➲ http://www.sqlite.org
SQLite levelezési lista
➲ http://groups.yahoo.com/subscribe/sqlite

Memóriakezelés haladóknak

A GNU C könyvtár néhány hasznos függvényének segítségével nemcsak takarékoskodhatunk az értékes memóriával, de csúnya hibákat is elháríthatunk.

A dinamikus memóriakezelés hagyományosan a C- és C++-programozás legkellemetlenebb része volt. Nem is meglepő, hogy bizonyos könnyebb, vagy legalábbis könnyebbnek tartott nyelvek, például a Java a szemétgyűjtő eljárások révén megszabadította a programozókat ettől a tehertől. A nehézsúlyú C-programozók számára azonban a GNU C könyvtár néhány olyan eszközt kínál, amelyek segítségével a memóriahasználat kézben tartható, ellenőrizhető és követhető.

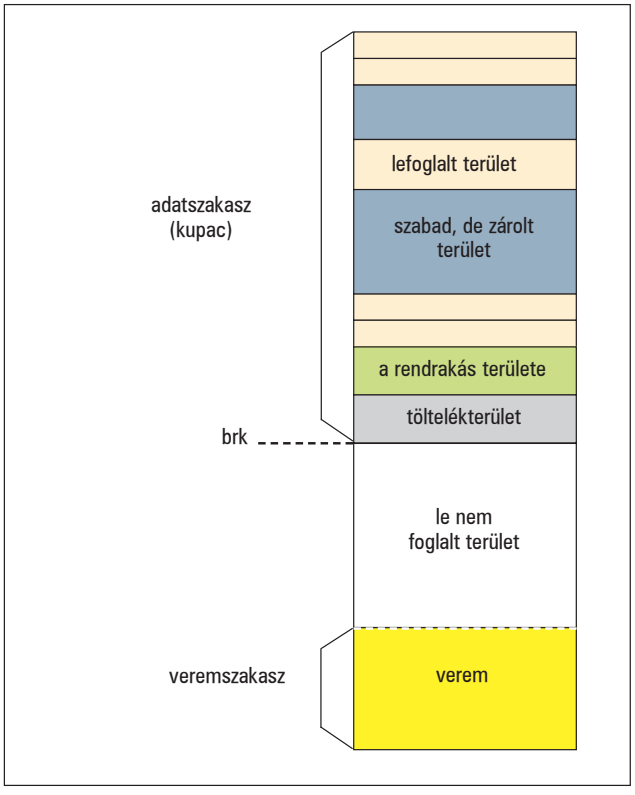
A memóriakezelés alapjai

Az adott folyamathoz tartozó memóriaterület általában lehet statikus, ekkor méretének meghatározására a fordításkor kerül sor, illetve dinamikus, ekkor a szükséges terület nagysága futás közben dől el. Az utóbbi esetben a memória két részre osztható: a kupacra (heap), ami a `malloc()` függvénnyel foglalt területeket öleli fel, illetve a veremre (stack), ami a függvények ideiglenes munkaterületét jelenti. Mint *ábránk* is szemlélteti, a kupac felfelé, míg a verem lefelé növekszik.

Ha egy folyamatnak memóriára van szüksége, akkor a `brk()` vagy az `sbrk()` rendszerhívással lehet kitolni a kupac felső határát. Mivel a rendszerhívások a processzorterhelés szempontjából költségesnek tekinthetők, egyetlen `brk()` hívással inkább egy nagyobb területet érdemes lefoglalni, majd azt – szükség szerint – kisebb darabokra osztva felhasználni. A `malloc()` pontosan ezt teszi: nagyszámú kisebb `malloc()` kérést kevesebb, de nagyobb `brk()` hívásba fog össze. Ezzel a módszerrel számottevő teljesítménynövekedést lehet elérni. A `malloc()` hívás maga is jóval olcsóbb, mint a `brk()`, ugyanis könyvtári (library call), és nem rendszerhívás. Hasonló folyamatok játszódnak le, amikor a folyamat memóriát szabadít fel. A memóriablokkokat a rendszer nem kapja vissza azonnal, hiszen ehhez minden alkalommal – negatív átadott értékkel – egy `brk()` hívást kellene végrehajtani. Ehelyett a C könyvtár addig gyűjtöget, amíg elegendően nagy memóriaterület nem szabadítható fel egyetlen lépésben.

A nagyon nagy területre irányuló kérések kezeléséhez a `malloc()` az `mmap()` rendszerhívással keres megcímezhető memóriaterületet. Ezzel részben elháríthatók a memóriatöredezés nemkívánatos hatásai, amikor nagyobb memóriablokkokat szabadítunk fel, ám a felhasználásukat akadályozhatják a közéjük ékelődő és a lefoglalt terület végére kerülő, újabban lefoglalt, kisebb méretű darabok. Ilyenkor a `brk()` hívással korábban lefoglalt blokkot hiába szabadítja fel a folyamat, a rendszer továbbra sem tudja használni.

A dinamikus memóriakezeléssel kapcsolatos függvények között a `malloc()` és a `free()` mellett számos továbbit is találunk, ám ezt a kettőt használjuk a legtöbbet. Érdemes megemlíteni a `realloc()` függvényt, amivel már lefoglalt blokkot tudunk átméretezni; a `calloc()` függvényt, ami „kitisztított” területet foglal; a `memalign()`, `posix_memalign()` és `valloc()` függvényt, amelyek egybefüggő terület foglalására alkalmasak.



A kupac és a verem egymással ellentétes irányban növekedik

A memóriaállapot kezelése

A C könyvtár memóriakezelő kódrészlete alapvetően általános jellegű memóriakezelésre való. Ez a megközelítés az esetek túlnyomó részében megfelelő teljesítményt eredményez, ám bizonyos programoknál érdemes módosítani a beállításokon. Először is a `malloc_stats()` vagy a `mallinfo()` könyvtári hívással gyűjtsünk adatokat a memóriahasználatról. Az előbbi tömör összefoglalót küld a program memóriahasználatáról a szabványos hibakimenetre. Az összefoglaló tartalmazza a rendszertől `brk()` hívások által lefoglalt bájtok számát, a `malloc()` hívásokkal ténylegesen használatba vett terület nagyságát, illetve az `mmap()` hívásokkal igényelt memóriaméretet. Példa a hívás kimenetére:

```
Arena 0:
system bytes     =    205892
(rendszerbájtok száma)
in use bytes     =    101188
(használatban lévő bájtok száma)
Total (incl. mmap):
(összes, az mmap hívásokkal együtt)
system bytes     =    205892
(rendszerbájtok száma)
in use bytes     =    101188
(használatban lévő bájtok száma)
```

```
max mmap regions     =    0
max mmap regions     =    0
```

Ha pontosabb adatokat akarunk látni, és nem csak egy egyszerű kiíratást szeretnénk, a `mallinfo()` hívást használhatjuk. A függvény egy `mallinfo` nevű adatszerkezetet ad vissza, ami különféle, a memóriaállapottal kapcsolatos állapotjelzőket tartalmaz. Ezek közül a legfontosabbak leírását a „A legfontosabbak a `mallinfo` által visszaadott adatok közül” című széljegyzet tartalmazza. Az adatszerkezet teljes körű ismertetése a */usr/include/malloc.h* című állományban található meg.
A `malloc_usable_size()` a libc egy másik hasznos függvénye, ami egy korábban lefoglalt memóriaterületből felhasználható bájtok számát adja vissza. A minimális méretekre vonatkozó megszorítások és a területek folytonosságára való törekvés miatt ez a mennyiség az eredetileg kértnél nagyobb is lehet. Ha például lefoglalunk 30 bájtot, a ténylegesen felhasználható terület 36 bájt lesz. Vagyis legfeljebb 36 bájtot írhatunk be a memóriablokkba úgy, hogy más blokkokat ne írjunk felül. Természetesen ennél visszataszítóbb programozási módszert nehéz lenne említeni, és erősen változatfüggő is, tehát semmiképpen ne használjuk. A `malloc_usable_size()` elsősorban mint hibakereső eszköz lehet hasznos. Alkalmas például arra, hogy ellenőrizzük egy kívülről kapott memóriablokk méretét, még mielőtt beleírnánk valamit.

A foglalási stratégia

A memóriakezelő függvények viselkedését bizonyos beállítások módosításával, a `mallopt()` függvény segítségével lehet módosítani (*1.* és *2. lista*).
A függvény prototípusa, illetve négy alapvető, általa módosítható beállítás az SVID/XPG/ANSI szabványban is szerepel.
A jelenlegi GNU C könyvtári megvalósítás (a 2.3.1-es változat a cikk születésének időpontjában) ezek közül csak egyet támogat (`M_MXFAST`), a többi hármat nem. Másrészről a könyvtár további négy, a szabvány által nem említett beállítással vigasztal minket. A `mallopt()` segítségével módosítható beállítások leírása az „A `mallopt()` segítségével módosítható beállítások” című széljegyzetben található.
A foglalás hangolása a `mallopt()` hívásoknak a programba való építése és a program újrafordítása nélkül is megoldható. Ez különösen akkor hasznos, ha bizonyos értékeket gyorsan ki akarunk próbálni, vagy ha nem rendelkezünk a forráskóddal. Mindössze annyit kell tennünk, hogy a program futtatása előtt beállítjuk a megfelelő környezeti változót. *Táblázatunk* a `mallopt()` beállítások és a környezeti változók egymásnak való megfeleltetését tartalmazza, némi plusztájékoztatással kiegészítve. Ha például a „rendrakási” határértéket 64 KB-ra akarjuk állítani, az alkalmazást így kell indítani:

```
MALLOC_TRIM_THRESHOLD=65536 alkalmazás
```

A rendrakásról szót ejtve: a `malloc_trim` (töltelék) hívással lehet elindítani a memória rendbetételét és a használaton kívüli területek rendszernek való visszaadását. A függvény átméretezi az adatszakaszt, legalább ***töltelléknyi*** részt hagyva a végén, és sikertelenül tér vissza, ha nem tudott legalább egy lapnyi területet felszabadítani. A szegmensméret mindig a lapméret – ez i386 processzornál 4096 bájt – többszöröse. A rendrakásra elérhető memória méretét a `mallinfo()` által visszaadott adatszerkezet `keepcost` összetevője tartalmazza. Önműködő rendrakást a `memory_trim()` meghívásával a `free()` függvény is végez, feltéve, hogy a `keepcost` értéke nagyobb az `M_TRIM_THRESHOLD` értékénél, átadott töltelék értékként pedig az `M_TOP_PAD`-et veszi.

A legfontosabbak a „mallinfo()” által visszaadott adatok közül

- `arena`: a kupacból lefoglalt terület teljes mérete (vagyis az azt jelző mérőszám, hogy a határpont mennyivel mozdult el a folyamat indítása óta).
- `uordblks`: a lefoglalt és használatban lévő bájtok száma.
- `fordblks`: a lefoglalt, de használatban nem lévő bájtok száma.
- `keepcost`: a `malloc_trim()` hívása révén felszabadítható terület nagysága.
- `hblks`: az `mmap()` segítségével lefoglalt darabok száma.
- `hblkhd`: az `mmap()` segítségével lefoglalt bájtok száma.

Memóriával kapcsolatos hibakeresés: az éppség ellenőrzése

A memóriakezeléssel kapcsolatos hibák keresése az egyik leginkább időtrabló dolog az összetett alkalmazások fejlesztésekor. Ilyenkor egyrészt a blokkok lefoglalását és felszabadítását kell tisztába tenni, másrészt fel kell deríteni a memóriakezelési hibákat.
A memória sértetlensége akkor bomlik meg, ha olyan helyre írunk, ami ugyan az adatszakaszban található, de az eredetileg használni kívánt memóriablokk határain kívülre esik. Jó példa erre az, amikor egy tömb végén túlra írunk adatot. Természetesen, ha az adatszakaszon kívülre próbálnánk írni, akkor a program szakaszolási (segmention) hibával azonnal leállna, illetve a megfelelő jelzéskezelő meghívására kerülne sor, és így azonosítani lehetne a hibát okozó utasítást. A sértetlenség megbomlása ennél kellemetlenebb, mivel észrevétlenül is lezajlódhat, és később a hibás résztől akár messzeeső programrészekben okozhat rendellenes működést. Minél előbb fedezzük fel tehát a hibát a programban, annál nagyobb eséllyel tudjuk elhárítani a következményeit.
Az adatépség megbomlása más memóriablokkokat is érinthet, és nemcsak az alkalmazás adatainak, de a kupackezelő adatszerkezeteknek a sérülését is okozhatja. Az előbbi esetben a hibára egyedül saját adatszerkezeteink tartalmának az elemzésével deríthetünk fényt. Az utóbbi esetben a GNU libc sértetlenséget ellenőrző eszközeire támaszkodhatunk, amelyek értesítenek minket, ha valamilyen hiba merül fel.
Adott programon belül a memória ellenőrzése önműködően és kézzel egyaránt végezhető. Az előbbi a `MALLOC_CHECK_` környezeti változó értékének megadásával lehetséges:

```
MALLOC_CHECK_=1 alkalmazás
```

1. táblázat A mallopt() beállítások és a környezeti változók megfeleltetése

mallopt()-beállítás	környezeti változó	alapérték	megjegyzés
M_TRIM_THRESHOLD	MALLOC_TRIM_THRESHOLD_	128 KB	-1U letiltja
M_TOP_PAD	MALLOC_TOP_PAD_	0	
M_MMAP_THRESHOLD	MALLOC_MMAP_THRESHOLD_	128 KB	0 letiltja
M_MMAP_MAX	MALLOC_MMAP_MAX_	64	0 letiltja

1. lista A rendrakás határértékének beállítása a mallopt() függvénnyel

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

int main(int argc, char *argv) {

  int thr;
  char *p1;

  if (argc != 2) {
      printf("Használat: 1_kódrészlet
      ➥<rendrakás határérték " "[KB]>\n");
      exit(0);
  }

  thr = atoi(argv[1])*1024;
  if (!mallopt(M_TRIM_THRESHOLD, thr)) {
      printf("sikertelen mallopt()\n");
  }

  printf("100 k lefoglalása, a rendrakás
  ➥határértéke " " %d bájt\n", thr);
  p1 = malloc(100000);
  malloc_stats();

  printf("\n100 k felszabadítása\n");
  free(p1);
  malloc_stats();
}
```

Ezzel a módszerrel a határtúllépések jelentős része felfedezhető, és egyes esetekben a program összeomlása is megelőzhető.
A hiba felismerésekor végrehajtott művelet a `MALLOC_CHECK_` értékétől függ: 1-es értéknél a rendszer hibaüzenetet küld a `stderr` kimenetre, de magát a programot tovább engedi futni; 2-es értéknél kimenet nélkül leállítja a programot; 3-as értéknél az 1-es és 2-es eset hatása összeadódik.
Az önműködő ellenőrzés csak akkor játszódik le, amikor memóriakezeléssel kapcsolatos függvényeket hívunk meg. Ha tehát egy tömbbe a határain túlnyúlva írunk, akkor csak a következő `malloc()` vagy `free()` hívásnál fogunk hibaüzenetet kapni. A hibák egy része sajnos észrevétlen marad, és a hibaüzenetek tartalma sokszor – finoman szólva – kevésbé hasznos. Egy `free()` hívásnál például tudjuk, hogy melyik mutatót szabadítottuk fel a hiba felismerésekor, de azt nem, hogy pontosan mi vagy ki zavarta össze a kupac tartalmát. Ha lefoglalás közben derül ki valamilyen hiba, a „heap corrupted” (kupac sérült) üzenetet kapjuk.
Egy másik lehetőség az, hogy ellenőrzési pontokat helyezünk el a programban. Ilyenkor még a program futásának megkezdésekor meg kell hívnunk az `mcheck()` függvényt. Segítségével olyan egyedi hibakezelőt vehetünk használatba, amelyet minden egyes kupachiba felismerésekor meg tudunk hívni. Egy alapértelmezett hibakezelő is rendelkezésünkre áll arra az esetre, ha nem írtunk sajátot. Ha az `mcheck()` meghívásán túlestünk, a `MALLOC_CHECK_` révén elérhető, sértetlenséget ellenőrző lehetőségek máris rendelkezésünkre állnak. Emellett az `mprobe()` függvényt bármikor meghívhatjuk, ha a megadott memóriamutatót szeretnénk ellenőrizni. Az `mprobe()`

2. lista Kisebb rendrakási határértéket megadva memóriát takaríthatunk meg

```
branzo@betelgeuse:~/malloc_debug$
➥./1_kodreszlet 128

100 k lefoglalása, a rendrakás határértéke
➥131072 bájt
Arena 0:
system bytes =      103724
in use bytes =      100012
Total (incl. mmap):
system bytes =      103724
in use bytes =      100012
max mmap regions    =           0
max mmap bytes      =           0

100 k felszabadítása
Arena 0:
system bytes =      103724
in use bytes =           4
Total (incl. mmap):
system bytes =      103724
in use bytes =           4
max mmap regions    =           0
max mmap bytes      =           0

branzo@betelgeuse:~/malloc_debug$
./1_kodreszlet 64

100 k lefoglalása, a rendrakás határértéke
➥65536 bájt
Arena 0:
system bytes =      103724
in use bytes =      100012
Total (incl. mmap):
system bytes =      103724
in use bytes =      100012
max mmap regions    =           0
max mmap bytes      =           0

100 k felszabadítása
Arena 0:
system bytes =        1324
in use bytes =           4
Total (incl. mmap):
system bytes =        1324
in use bytes =           4
max mmap regions    =           0
max mmap bytes      =           0
```

Az mprobe() visszatérési értékei

- `MCHECK_DISABLED`: sértetlenség ellenőrzése kikapcsolva (az `mtrace()` meghívására nem került sor).
- `MCHECK_OK`: a blokk rendben van.
- `MCHECK_FREE`: a blokkot kétszer szabadítottuk fel.
- `MCHECK_HEAD`: a blokk előtti memóriaterület hibás.
- `MCHECK_TAIL`: a blokk utáni memóriaterület hibás.

A „mallopt()” segítségével módosítható beállítások

- `M_TRIM_THRESHOLD`: az a legkisebb memóriamennyiség, amelynek felszabadítása az adatszakasznak `brk()` hívással történő leszűkítését váltja ki a memóriakezelő rendszerből. A felszabadított adatterület – mint már esett róla szó – a rendszerhívások számának csökkentése érdekében valójában nem kerül vissza azonnal a rendszerhez. Ez az érték adja meg, hogy mennyi memória felszabadítása után kerül sor egy ilyen hívásra. A rendrakási határérték nagyban befolyásolhatja a rendszer teljesítményét. Ha túl magasra állítjuk, akkor több memóriát tartunk vissza, és a rendszer nagyobb valószínűséggel veszi használatba a nagyságrenddel lassabb cseretárhelyet. Ellenben ha túlságosan kis értéket adunk neki, akkor a gyakori `brk()` hívások fogják a teljesítmény romlását okozni. A beállítás főként hosszú ideig futtatott programoknál fontos, ezek ugyanis nagy mennyiségű memóriát szívhatnak el a rendszertől. Azt is fontos megjegyezni, hogy a töredezettség miatt „zárolt” darabok nem adhatók vissza a rendszernek, mégha méretük nagyobb is a határértéknél – éppen ezért van szükség az `mmap()` függvénnyel végzett foglalásra. Az 1. kódrészletben található program a határértéket a neki átadott érték szerintire állítja, majd 100 KB memóriát foglal le és szabadít fel. Az eredmény a 2. kódrészletben látható.
- `M_TOP_PAD`: mennyi tartalékhelyet kell foglalni (meghagyni), amikor egy `brk()` hívással a kupac bővítésére (szűkítésére) kerül sor. Amikor egy `malloc()` hívás hatására a `brk()` veszi át a vezérlést, ennyi pluszterületet igényel a rendszertől. Hasonlóan, amikor egy `free()` hívás hatására negatív `brk()` hívás történik, ennyi pluszhelyet hagy meg a könyvtár jövőbeli használatra. Fontos, hogy az értékét csak megfontoltan változtassuk meg, ha a programunk nagy mennyiségű egymást követő `malloc()` és `free()` hívást hajt végre, nagy számú `brk()` hívást váltva ki ezzel.
- `M_TRIM_THRESHOLD`: az ennél nagyobb méretű memóriaterületre vonatkozó igények kielégítése `brk()` helyett `mmap()` hívással történik. Mint már utaltam rá, az utóbbi eljárás előnye az, hogy az így lefoglalt memóriaterület – a rendrakási határérték elérése nélkül is – felszabadítás után azonnal elérhető lesz a rendszer számára, ellentétben a `brk()` hívással lefoglalt részekkel. Ha kisebb értéket adunk neki, akkor több `mmap()` és kevesebb `brk()` hívás történik, és kevesebb lesz a zárolt memóriatöredék. Hátránya, hogy az `mmap()` általában lassabban intézi a memóriafoglalást, mint a `brk()` – az esetek nagy részében tehát a legjobb az, ha az alapértelmezett értéket hagyjuk meg.
- `M_MMAP_MAX`: az `mmap()` által egyszerre lefoglalható memóriadarabok maximális száma. Olyan rendszereknél van jelentősége, amelyeknél korlátozva van a folyó `mmap()` eljárások száma. Általában az a legjobb, ha nem nyúlunk hozzá.
- `M_MXFAST`: a fastbin memóriakonténerek használatával teljesített kérések által lefoglalható blokkok maximális mérete. Helyszűke miatt nem foglalkozhatunk részletesen a fastbin konténerekkel, annyit viszont érdemes tudni róluk, hogy elsősorban sok kicsi, azonos méretű blokk lefoglalásakor vagy felszabadításakor használhatók hatékonyan. Megjegyzem, mivel a fastbin konténerek soha nem kerülnek összevonásra (vagyis két szomszédos szabad fastbin soha nem egyesül egyetlen nagyobb, szabad blokkba), használatuk növeli a memória töredezettségét, illetve a program összesített memóriaszükségletét. Ebből következik, hogy az érték egyfajta megalkuvás eredménye a sebességnövekedés (kisebb fastbinek) iránti igény és a velük járó töredezés elkerülése (nagyobb fastbinek) iránti szándék között. A kisebb értékek az esetek jelentős részében megfelelők, ugyanis a GNU C könyvtárat úgy optimalizálták, hogy a nagyobb fastbinek jelentős töredezést okoznak, miközben használatuk nem jár érzékelhető sebességnövekedéssel. A fastbin-támogatás a `malloc` eljárások teljes újraírásakor, a 2.3-as változattal jelent meg a GNU C könyvtárban.

által visszaadott értékek ismertetése a „Az `mprobe()` visszatérési értékei” című széljegyzetben található.
Ha a teljes kupacot, és nem csupán egyetlen blokkot kívánunk ellenőrizni, az `mcheck_check_all()` meghívásával az összes aktív blokkon végigmehetünk. Arra is utasíthatjuk a memóriakezelő rutinokat, hogy csak a pillanatnyi blokk ellenőrzése helyett az `mcheck_check_all()` függvényt használják, ehhez induláskor az `mcheck()` helyett az `mcheck_pedantic()` függvényt kell meghívni. Azt azonban ne feledjük, hogy ezzel elég sok erőforrást kötünk le.
A memória-ellenőrzés engedélyezésének harmadik módja az alkalmazás libmcheckkel való fordítása:

```
gcc alkalmazás.c -o alkalmazás -lmcheck
```

Az `mcheck()` függvény így még az első memóriafoglalás végrehajtása előtt önműködően meghívásra kerül – ez különösen akkor érdekes, ha a program már a `main()` függvény elérése előtt is foglal dinamikus memóriablokkokat.

Memóriával kapcsolatos hibakeresés: a blokkok nyomon követése

A memóriablokkok történetét végigkövetve könnyebben felismerhetők a memóriaszivárgásokkal kapcsolatos gondok, illetve a már felszabadított blokkok használata vagy újbóli felszabadítása. A GNU C könyvtár ilyen célokra nyomkövetési lehetőséget kínál, amit az `mtrace()` függvény meghívásával vehetünk igénybe. A hívás végrehajtása után a rendszer minden kupacműveletet egy fájlba naplóz. A fájl nevét a `MALLOC_TRACE` környezeti változóban kell megadni. A naplófájl elemzését egy Perl-parancsfájllal később is el lehet végezni, ami szintén a könyvtár része, és – nem meglepő módon – szintén az ***mtrace*** nevet viseli. A naplózás a `muntrace()` függvény hívásával állítható le, de ne feledjük, ha a programnak csak bizonyos részeire végezzük el a naplózást, akkor az utólagos feldolgozáskor kapott eredmények teljesen értelmetlenek is lehetnek. Nem létező szivárgást fedezhetünk fel például úgy, hogy lefoglalunk egy memóriablokkot, majd a `muntrace()` meghívása után szabadítjuk fel.
Az alábbiakban a nyomkövetésre látható egy példa a *3. listában* szereplő programmal:

```
$ gcc -g 3_kodreszlet.c -o 3_lista
$ MALLOC_TRACE="nyomkovetes.log" ./3_lista
$ mtrace nyomkovetes.log

Memory not freed: (Fel nem szabadított memória)
-----------------
```

```
Address (Cím)    Size (Méret)     Caller (Hívó)
0x08049718       0xa  at
malloc_debug/3_lista.c:9
```

A memória nyomkövetése nem befolyásolja a hibák kialakulását, így attól még, hogy az `mtrace()` meghívása megtörtént, a program akár össze is omolhat. Ennél csak az a rosszabb, ha a program szakaszolási hibát követ el, hiszen ekkor a nyomkövetés naplófájlja befejezetlen marad, és akár ellentmondásos adatokat is tartalmazhat. Ez ellen úgy védekezhetünk, hogy egy SIGSEGV jelzéskezelőt készítünk, ami a `muntrace()` függvényt hívja meg, és ezáltal még a program leállása előtt gondoskodik a naplófájl lezárásáról (*4. lista*).
A memória nyomkövetéséről a libc infooldalán található további tájékoztatást.

Mélyebbre merülve

Megeshet, hogy a GNU C könyvtár normál hibakeresési lehetőségei egy program esetében elégtelennek bizonyulnak. Ebben az esetben külső eszközt is használhatunk a memória-kezelési hibák felderítésére (lásd a *listákon*), de mélyebbre is beáshatjuk magunkat a könyvtár rejtelmeibe. Ehhez csupán három függvényt kell megírnunk, majd rá kell aggatni őket a következő előre megadott változókra:

- `__malloc_hook` – arra a függvényre mutat, ami akkor jut szerephez, ha a felhasználó a `malloc()` függvényt hívja meg. Ebben tetszőleges ellenőrzéseket és számlázást (accounting) végezhetünk, majd a valódi `malloc()` meghívásával lefoglalhatjuk a kért memóriaterületet.
- `__free_hook` – a normál `free()` helyett meghívandó függvényt adja meg.
- `__malloc_initialize_hook` – a memóriakezelő rendszer üzembe helyezésekor meghívott függvényre mutat. Így mód nyílik arra, hogy bizonyos műveleteket hajtsunk végre, például még a többi memóriakezelő művelet elindulása előtt megadhatjuk az előbbi két változó értékét.

3. lista Nyomkövetés az mtrace() segítségével

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

int main() {
  char *ptr;

  mtrace();
  ptr = malloc(10);
  /*  free(ptr); */

}
```

4. lista A SIGSEGV kezelőben meg kell hívni a muntrace() függvényt

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

void handler(int s) {
  muntrace();
  abort();
}

int main() {
  char *ptr;

  signal(SIGSEGV, handler);
  mtrace();
  ptr = malloc(10);
  free(ptr);
  free(ptr);

}
```

Hasonló módosítási lehetőségünk van a többi memóriakezelő függvény esetében is, mint a `realloc()`, `calloc()` stb. Ügyeljünk arra, hogy mentsük a változók korábbi értékeit, majd a `malloc()` vagy `free()` fenti eljárásainkban történő meghívása előtt visszaállítsuk. Ha ezt elmulasztjuk, a program végtelen önhívó ciklusba esik, és használhatatlan lesz. Érdemes egy pillantást vetni a libc infooldalán található memória-hibakeresési példában található kódra, az összes fontos részlet kiderül belőle.
Utolsó megjegyzésként annyit még megemlítenék, hogy a fenti változókat az mcheck- és az mtrace-eszközök is használják. Ha tehát mindet egyszerre akarjuk igénybe venni, nagyon nagy körültekintésre lesz szükségünk.

Összegzés

A GNU C könyvtár számos bővítménnyel rendelkezik, amelyek roppant hasznosnak bizonyulnak a memóriakezeléssel kapcsolatos kérdések megoldásában. Ha finomhangolni szeretnénk egy alkalmazás memóriahasználatát, vagy a saját igényeinkre szabott memória hibakereső megoldást akarunk készíteni, akkor minden bizonnyal jól tudjuk használni ezeket az eszközöket, és megfelelő kiindulópontként szolgálnak saját eljárásaink kidolgozásához.

Linux Journal 2003 május, 109. szám

Gianluca Insolvibile (g.insolvibile@cpr.it)
A 0.99pl4-es rendszermag óta Linux-rajongó. Jelenleg hálózati és digitális videokutatással és -fejlesztéssel foglalkozik.

KAPCSOLÓDÓ CÍMEK

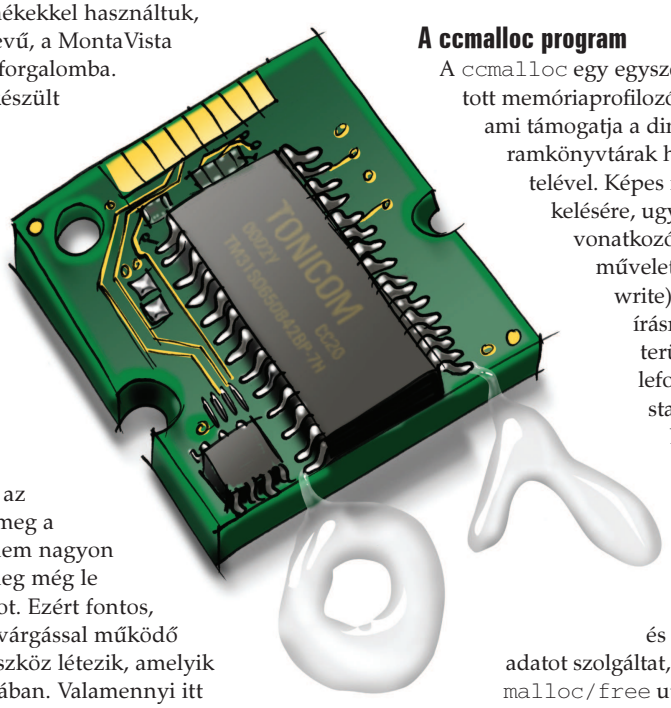
„Debugging Memory on Linux”, írta Petr Sorfa, Linux Journal, 2001. július; elérhető a ➲ http://www.linuxjournal.com/article/4681 címen.
Electric Fence ➲ ftp.perens.com/pub/ElectricFence
A GNU C könyvtár kézikönyve ➲ http://www.gnu.org
mpatrol ➲ http://www.cbmamiga.demon.co.uk/mpatrol
NJAMDB ➲ http://sourceforge.net/projects/njamd

Memóriaszivárgás érzékelése C++-környezetben

Ne halogassuk a memóriaszivárgási gondok megoldását! Egy, esetleg több kényelmes eszközt is programfejlesztési folyamatunk részévé tehetünk.

Egyik korábbi cikkünk (Memóriaszivárgás érzékelése beágyazott Linux-rendszerekben, Linuxvilág, 2002. szeptember) a memóriaszivárgás érzékelésének kérdéseit tárgyalta C programozási nyelvet használó környezetben. Jelenlegi írásunk viszont a C++-programokban jelentkező memóriaszivárgás érzékelésével kapcsolatos kérdésekre keres választ, nem pedig a rendszermagbeli szivárgásokra. Valamennyi itt bemutatott eszközt a MontaVista Linux Professional Edition 2.1 és 3.0-s termékekkel használtuk, egyikük pedig, a `dmalloc` nevű, a MontaVista Linuxszal kerül kereskedelmi forgalomba.

A beágyazott rendszerekhez készült alkalmazói programok tervezőinek és programozóknak különös gondot kell fordítaniuk e programok erőforrás-felhasználására. A munkaállomásoktól eltérően a beágyazott rendszerek csupán véges tárterülettel rendelkeznek. Az ilyen rendszerekben nem található lapozási terület a működésüket szüneteltető programok számára. Amint a rendszer felhasználja az összes erőforrását, a pánikon meg a rendszer újraindításán kívül nem nagyon marad más választásunk, esetleg még le lehet állítani egy-két programot. Ezért fontos, hogy elkerüljük a memóriaszivárgással működő programokat. Számos olyan eszköz létezik, amelyik segít a szivárgások megtalálásában. Valamennyi itt bemutatott eszközhöz létezik saját tesztprogram.



Az egyik módszer sikeres alkalmazásfejlesztői használatát magam is láttam, ez a prototípuskód fejlesztéséhez és kipróbálásához készült, és amennyire csak lehetséges, egy munkaállomás használatát is magában foglalja. A munkaállomáson való kipróbálással az alkalmazói programok fejlesztői megkönnyíthetik a célprocesszorra való áttérést. A munkaállomások használata melletti döntő érv, hogy olcsók és minden érintett résztvevő birtokában van ilyen. Ezzel szemben a célgépek száma kevés, az iránt uk való kereslet pedig nagy.

A legtöbb memóriaszivárgás-érzékelő program teljes forráskód formájában érhető el. Ezeket jellemzően x86-alapú felületeken készítették. Nem x86-os felületeken történő használatuk új felületre való átültetést igényel (porting). Ez az átültetési művelet az egyszerű újrafordítástól, szerkesztéstől és futtatástól az egyik felületről a másiknak megfelelő gépi kód átalakításáig terjedhet. Némelyik eszközzel használati tanácsokat illetve javaslatokat kapunk a más felületekre történő programfordításhoz.

A dmalloc használata

Amint arról a 2002. szeptemberi számban már részletesen beszámoltam, a `dmalloc` szerzőjének állítása szerint a program C++-ismerete korlátozott, így a C++ alapú memóriaszivárgások felismerése úgyszintén az.

Annak érdekében, hogy a `dmalloc`-ot a C++-szal és programszálakkal együtt lehessen használni, az alkalmazást `static`-ként kellett szerkeszteni.

A ccmalloc program

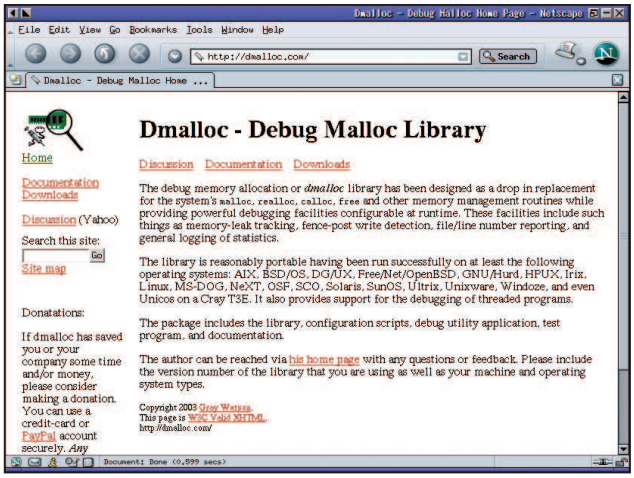
A `ccmalloc` egy egyszerű használati modellel ellátott memóriaprofilozó (memory-profiler) eszköz, ami támogatja a dinamikusan szerkesztett programkönyvtárak használatát, a `dlopen` kivételével. Képes memóriaszivárgások érzékelésére, ugyanazon adatállományokra vonatkozó többszörös felszabadítási műveletekre, hitelesítésre (underwrite), felülírásra (overwrite) és írásra (write) a már felszabadított területen. Megjelenít állomány-lefoglalási és -felszabadítási statisztikákat.

Egyaránt alkalmazható tökéletesített (optimized) és lecsupaszított (stripped) programkódra, és támogatja a C++ programnyelvet. A teljes hívási láncra vonatkozó állomány és programsor-számozási adatot szolgáltat, és nem egyes-egyedül a `malloc/free` utasítás közvetlen hívójára vonatkozóan, a C++ nyelv itt is támogatott.

A `ccmalloc` használatához nincs szükség a program újrafordítására: mindössze a szerkesztésnél (link) a `-lccmalloc -ldl` vagy a `ccmalloc.o -ldl` parancsokat kell használni. A `ccmalloc` képes a hívási láncolatokat hatékonyan ábrázolni, a hívási láncok testre szabható nyomtatását, hívási láncok kiválasztás szerinti nyomtatását, tömörített naplóállományt és a ***.ccmalloc*** névvel ellátott indítóállományt. A leírás fő része a ***ccmalloc.cfg*** nevű állományban található, a fejlesztőeszközhöz tartozó próbaállományok további leírást tartalmaznak. A rendszerváltozók eléréséhez az `nm` és `gdb` szükséges, míg a gzip a naplóállományok tömörítéséhez.

NJAMD

Az NJAMD, miként a program alkotója állítja, „nem csupán egy újabb malloc-hibakereső”. Mint az a legtöbb tárterület-foglalást ellenőrző hibakereső programnál megfigyelhető, a szabványos területfoglalási ellenőrző műveleteket újakkal helyettesítik, amik különféle ellenőrzéseket végeznek a tár használatbavétele során. Nevezetesen a program ellenőrzi a dinamikus



átmeneti tár alul-, illetve túlcsordulásait és érzékeli a tárterület felszabadítás utáni újbóli használatát. Az NJAMD-hez készített programkönyvtár LD_PRELOAD-olható vagy a programhoz szerkeszthető. Az első tárterület-foglaláskor nagy – 20 MB-os – tárhoz kapcsolódó átmeneti tárat hoz létre, és ebből szab ki annyi memóriát, amennyit a program igényel.
Az NJAMD használható önmagában, ügyféloldali programmal együtt, vagy a `gdb`-ből. Egy segédprogramja is van, ami a rendszerleállás utáni veremellenőrzést teszi lehetővé (postmortem heap analysis). Egy további szolgáltatás lehetővé teszi az alkalmazás hibakeresését, átugorva az újrafordítást: egyszerűen csak előre be kell tölteni a programkönyvtárat (preload).
Az NJAMD képes nyomon követni a szivárgásokat a `malloc` és `free` függvényeket burkoló könyvtári függvényekben, grafikus felhasználói felület (GUI) alapú helyfoglaló elemeknél, valamint a C++ `new` és `delete` függvényeknél.
Gyakran előfordul, hogy a memóriaszivárgás nem fedezhető fel azonnal, hanem lappang, hogy aztán később, a legváratlanabb pillanatban sújtson le. Ennek kiderítése sok időt vehet igénybe. Az NJAMD számos rendszerváltozóval bír, amelyek különböző érzékelési szintek beállítását teszik lehetővé.
Mint a legtöbb hibakereső eszköznél már megszokhattuk, a teljesítmény fontos kérdéssé válhat az NJAMD használatakor, ezért az eszközt leginkább csak a fejlesztési szakaszban szabad használni. A bekapcsolt állapotú eszköz mellett végzett telepítés a rendszer lelassulását eredményezheti.

YAMD

A YAMD, vagyis a Yet Another Memory Debugger szavakból képzett névvel ellátott program egy olyan újabb csomag, ami a lefoglalt tárterületek széleit vizsgálja. Ezt a processzor lapozási módszerének használatával végzi. A program képes érzékelni a határokon kívüli olvasási és írási műveleteket. A hibát a program már az azt előidéző utasításnál észleli, nem pedig később, amikor további hozzáférések jelentkeznek. A híváselfogásokat a program naplóban rögzíti, az állomány nevével, a programsor számával és nyomkövetési adatokkal együtt. A nyomkövetés azért fontos, mert a tárterület lefoglalása véges számú eljáráson keresztül történik meg.
A könyvtár a `malloc` és `free` hívásokat utánozza. E művelet végrehajtása sok közvetett `malloc` hívás elfogását jelenti, amilyeneket például a `strdup` kezdeményezett. Ezenkívül a `new` és `delete` műveleteket is elfogja, kivéve, ha azok túl vannak terhelve.
A YAMD, más rokon programokhoz hasonlóan, nagy mennyiségű látszólagos memóriát vagyis lapozási területet igényel a működéséhez, viszont beágyazott rendszeren ez működik. Ezt az eszközt munkaállomáson prototípus-hibakeresésre is használhatjuk. Ha ezzel a hibakereséssel végeztünk, az alkalmazás célprocesszorra történő ültetése során biztosak lehetünk abban, hogy a legtöbb vagy éppen az összes memóriaszivárgást megtaláltuk.
A YAMD a futtatáshoz egy héjprogramot, a `run-yamd`-t kínálja, ami a program végrehajtásának megkönnyítésére szolgál. A program bizonyos kiinduló állapotokból való helyreállításhoz több lehetőséget nyújt. Naplóállományt lehet készíteni, amikor az ellenőrzött program magtárterület-kiíratást (core dump) hajt végre. A hibakereső eszközt a YAMD által vezérelt programokban a hibák felderítésére lehet használni. Mindazonáltal hibák a hibakereső eszköz használata során akkor is jelentkezhetnek, ha a YAMD-t előre betöltjük, ahelyett, hogy statikusan össze lenne építve a programmal.

Valgrind

A Valgrind viszonylag új, nyílt forrású tárterület-hibakereső az x86-os felületen használható GNU/Linux-rendszerekhez. Több képességgel van megáldva, mint a korábbi eszközök, viszont kizárólag x86-os gépeken fut. Ha egy program végrehajtása a Valgrind alatt zajlik, akkor az összes tárból való olvasási vagy odaírási művelet az összes `malloc`, `free`, `new` és `delete` eljárás ellenőrzésével zajlik. A Valgrind képes érzékelni a használatlan tárterületet, a memóriaszivárgásokat, az előkészítetlen memóriára vagy rosszul címezhető tárterületre való hivatkozást, a Posix-szálak hibás használatának néhány fajtáját, és a `malloc-free` és a `new-delete` műveletpárok típushibáit.
A Valgrind a hibaszűrésre a `gdb`-vel együtt is használható, ami lehetővé teszi a programozó számára, hogy a hiba jelentkezésének helyén a `gdb`-t használja. Ilyen módon a programozó közvetlenül a hibaforrást nézheti meg, és hamarabb találhat rá megoldást is. Némelyik esetben javítófolt is készíthető, a hibakeresés pedig folytatódhat. A Valgrindet úgy tervezték, hogy kicsi és nagy alkalmazásokon egyaránt képes legyen dolgozni, például a KDE 3-on, a Mozillán és az OpenOffice-on.
A Valgrind egyik szolgáltatása az, hogy képes részletes adatokat szolgáltatni az átmeneti tár használatának jellegéről (profiling). A program képes a processzor (CPU) L1-D, L1-I és egyesített L2 átmeneti tárának részletekbe menő utánzására, és a felhasználási eredményeket a vizsgált program minden egyes sorára vonatkozóan ki tudja számítani. A Valgrindnak jó a HOGYAN-ja, amelyben rengeteg példa található. A Valgrind honlapján temérdek leírás szerepel, bejárása mégis egyszerű. Számos különféle lehetőséget magában foglaló összeállítás érhető el, és a felhasználókra van bízva, hogy a számukra legkedvezőbbet kiválasszák.
A Valgrind hibamegjelenítő képernyőjén szerepel a vizsgált program azonosítója (PID), amit a hiba leírása követ. A címekkel együtt megjelennek a forrásállományok nevei és a programsorok számai. A teljes visszakeresés végig megjelenítődik.
A Valgrind működésének megkezdésekor beolvas egy indítóállományt, ami olyan utasításokat tartalmazhat, amelyek megmondják, hogy milyen hibaellenőrző üzenetek ne jelenjenek meg. E lehetőség használatával mód nyílik rá, hogy figyelmünket egy meghatározott programra összpontosítsuk, és ne a programkönyvtárakra, amiket különben sem lehet módosítani.
A Valgrind úgy végzi az ellenőrzést, hogy az alkalmazást szimulált processzorkörnyezetben futtatja. Ez a dinamikus szerkesztőt, illetve betöltőt (linker/loader) arra kényszeríti, hogy elsőként a szimulációs programot töltse be, s csak azután tölti

be a hozzátartozó könyvtárakat a szimulátorprogramba. A program futásának idején mindenféle adat gyűjtése zajlik. Amint a program végrehajtását megszakítjuk, a naplózott adatok megjeleníthetők, vagy akár kiírhatók a naplóállományba.

mpatrol

Az mpatrol programkönyvtár a tárterület lefoglalásának ellenőrzésére a vizsgálni kívánt programmal összeszerkeszthető. A program olyan módon lett megírva, hogy több különböző operációs rendszer felületén is futtatható. Határozottan nagy előnye, hogy több különböző processzorra át lett ültetve (porting), vagyis MIPS-re, PowerPC-re, x86-ra és néhány MontaVista fogyasztó által a StrongARM célgépekre.
Az mpatrol nagymértékben testreszabható: beállítható, hogy a verem használata helyett rögzített méretű statikus tömbből foglaljon le területeket. Statikus, osztott és szálbiztos könyvtárként lehet összeépíteni. Ugyanakkor a program megjelenhet egyetlen nagyméretű tárgykódú állomány formájában is, amit az alkalmazással össze lehet építeni, nem pedig könyvtárban kell elhelyezni. Ez a működési sajátosság a felhasználónak nagyfokú rugalmasságot biztosít.
A program által létrehozott kód negyvennégy különböző tárterület-foglalási és lánckezelési művelet pótlását tartalmazza. A kampók (hooks) adottak, így ezek az eljárások közvetlenül a `gdb`-ből hívhatók. Ez az adottság teszi lehetővé az mpatrolt használó programok számára a hibák keresését.
A könyvtárbeállítások és a veremhasználat a program működése során időszakosan megjeleníthető. A program futása idején gyűjtött összes statisztikai adat a program leállításakor megjelenik. A program előre beállított értékekkel rendelkezik, amelyek a környezeti változókkal felülbírálhatók. E változók futásidőben történő megváltoztatása szükségtelenné teszi a könyvtár újraépítését. A különféle tesztek hangolása dinamikusan végezhető el. Valamennyi naplózási tevékenység eredménye a pillanatnyi könyvtárban elhelyezkedő állományokba gyűjthető, de átirányítható a `stdout`-ra, a `stderr`-re vagy egyéb állományokba.
A program futása közben gyűjteni lehet a veremhívási visszakeresési adatot, és ezt állományba is lehet rögzíteni. Amennyiben a programot és járulékos könyvtárait a változók és programsorszámozás bekapcsolásával építették össze, ezeket az adatokat a naplóállományban ugyancsak meg lehet jeleníteni. Ha a programozó egy bizonyos ponton úgy dönt, hogy a terhelési tesztet egy kisebb tártartományon (smaller memory footprint) szeretné kipróbálni, az mpatrolt a tárterület korlátozására lehet utasítani. Ez a lehetőség olyan próbafeltételek megteremtését jelenti, amiket géptermi körülmények között készen talán sehol sem lehet megtalálni. A felhasználói környezetben végzett terhelési próba vagy egy gép próbaüzemeltetése ezzel a szolgáltatással könnyebbé válik. A fentebb említetteken kívül a hibajavítási eljárások ellenőrzése érdekében a próbaprogrammal tárterületfoglalási hibákat is elő lehet idézni. A programnak eme képessége hasznosnak bizonyulhat a C++ nyelvű kivételkezelésben. A veremről pillanatfelvételek készíthetők, hogy képet kaphassunk a tárhasználat alsó és felső értékeiről.

Insure++

A Parasoft cég által készített Insure++ termék nem tartozik a GPL-esített programok közé és nem is ingyenes, viszont jól használható eszköz a memóriaszivárgás és a kódlefedettség érzékelésére – nagyon hasonlít az mpatrolra.
Az Insure++ azonban az mpatrolnál ténylegesen több munkát végez a kódlefedettség területén, valamint adatgyűjtő és megjelenítő eszközökkel is szolgál. A program próbapéldányai letölthetők, és meghatározott ideig nem linuxos munkaállomásokon használhatók.
A program könnyen telepíthető Linuxra, de csomóponthoz kötött azon a gépen, amelyre telepítették. Az Insure++ programmal együtt átfogó leírást és számos bővítési lehetőséget kaphatunk. A kódlefedettséget vizsgáló eszköz különálló program, de már az alapcsomagban is szerepel.
Az Insure++ sok adattal szolgál a feltárt hibákról. Az Insure++ programot a használatbavételhez a felhasználói felülettel együtt kell újrafordítani. Ez a felhasználói program képezi az Insure++ könyvtári eljárásainak használatához szükséges programkódot. A fordítási szakaszban az érvénytelen típusátalakítások és a hibás értékátadások egyaránt felszínre kerülnek. A nyilvánvaló tárhasználati rendellenességeket ugyancsak jelenti a program. Futási időben a hibák a `stderr`-re kerülnek, de grafikus eszközzel is megjeleníthetők. Alkalmazás készítésekor a parancssor vagy a `makefiles` használható, ami nagy méretű alkalmazások létrehozását és komoly feladatok megoldását teszi lehetővé.
A program végrehajtása egyszerű: az Insure++ semmilyen különleges végrehajtandó parancsot nem igényel, úgy működik, mintha egy közönséges program lenne. Valamennyi hibakereső- és elfogó programkód az Insure++ programkönyvtáraiban található, amelyek össze lettek szerkesztve a programmal.
Az Inuse nevű bővítmény valós időben mutatja meg, hogyan használja a program a tárat. Képes pontos képet rajzolni a memóriafelhasználásról, hogy az mennyire válik töredezetté, és az apró, ám idővel egyre jelentősebbé váló kódszivárgásról. Egy ügyfélnél tapasztaltam, hogy egy bizonyos C++-osztálynál kis területű memória szivárgott, ami egy munkaállomásról nézve egészen aprónak látszott. Egy várhatóan hónapokig, sőt évekig futó beágyazott rendszeren a szivárgás meglehetősen felduzzadhat. Ezzel az eszközzel a szivárgásnak gyorsan nyomára bukkanhatunk, megtalálhatjuk és kijavíthatjuk. Más hozzáférhető eszközök még nem voltak képesek ezt a szivárgást észrevenni.
A kódlefedettséget egy további eszköz, a TCA vizsgálja. Amikor a program az Insure++ működése közben fut, adatokat gyűjt, és amint a TCA elemzi azokat, pontosan megmutatja, hogy milyen programkód lett végrehajtva. A TCA grafikus felhasználói felülete finomítja a kódlefedettségről alkotott képet.

Linux Journal 2003. június, 110. szám

Cal Erickson (cal_erickson@mvista.com)
Vezető Linux-tanácsadóként dolgozik a MontaVista Software cégnél. Ezt megelőzően a Mentor Graphics Embedded Software Division vezető tanácsadó mérnöke volt. Cal több mint harminc éve dolgozik a számítógép-ipar területén.

KAPCSOLÓDÓ CÍMEK

dmalloc ➲ http://dmalloc.com
ccmalloc ➲ http://www.inf.ethz.ch/personal/biere/projects/ccmalloc
Insure++
➲ http://www.parasoft.com/products/insure/index.htm
mpatrol ➲ http://www.cbmamiga.demon.co.uk/mpatrol
NJAMD ➲ http://sourceforge.net/projects/njamd
Valgrind ➲ http://developer.kde.org/~sewardj
YAMD ➲ http://www3.hmc.edu/~neldredge/yamd

Az eVote elektronikus szavazatszámláló program

A levelezőlisták remekül használhatók a gondolatok cseréjére, sőt mostantól akár szavazatszámlálásra is igénybe vehetők.

A konferenciarendszerrel ötvözött különleges adatbázis-kiszolgáló képes a valódi nem rangsorelvű demokráciák (nem hierarchikus demokrácia – erről ***Az eszményi demokrácia*** című részben olvashatsz részletesebben) számára a közvetítő közeget megteremteni. A szavazórendszer felügyeletét jelentő munka a felhasználókra és a programra hárul, nem pedig a rendszergazdára. Amikor a megoldást kínáló program hozzám került, nem tudtam az elektronikus demokráciát továbbra is csupán hirdetni – nekifogtam a megépítésének. A Clerk nevet viselő különleges szavazókiszolgáló program C++ nyelven készült, és számos szolgáltatással áll a szavazók rendelkezésére. Bárki kiírhat szavazást, és a felhasználók mindaddig megváltoztathatják véleményüket, amíg a szavazás le nem zárult. A szavazások lehetnek nyilvánosak (`public`), vagyis ekkor a szavazásban résztvevők láthatják egymás szavazatait; lehetnek titkosak (`private`); vagy az `if_voted` (szavazott-e) típusba sorolhatók. Ez utóbbi esetben megtudható, hogy kik adták le szavazatukat, az viszont nem, hogyan szavaztak. A program többféle szavazástípust is támogat: igen/nem (yes/no), számszerű és csoportos. A program könnyen bővíthető új szolgáltatásokkal és választástípusokkal a már létező osztályok kibővítése révén. A Clerk az adatok karbantartását röptében végzi, semmiféle segítséget nem igényelve a rendszergazdától. Az objektumközpontú programszerkezet ígéretéhez híven minden egyes új szolgáltatás hozzáadása fokozta a program munkabírását (robust).

A szavazócédulák

A szavazócédulák dinamikusak. Amint a felhasználó bezárja a szavazatleadási lehetőséget, egyúttal kiveszi a konferenciából; ha pedig a program más módon szünetelteti a tevékenységét, akkor a választás összeomlik, a memória felső vége felé görgetve a többi elemet, és helyet készítve további közvélemény-kutatások számára. Az elemobjektumok kiszámítják a helyüket a választási listában, és a program olyan módon van beállítva, hogy minden a rendszergazda bármiféle beavatkozása nélkül folytatódjék.

A Clerk és a szavazó közötti adatcsere

A `Clerk main()` tagfüggvény tulajdonképpen egy végtelen ciklus, ami az eVote-ügyfelektől a folyamatközi adatcsere-üzenetszolgáltatáson (interprocess communication messaging) keresztül beérkező üzenetekre figyel, vagyis a valódi felhasználókkal kapcsolatot tartó felhasználói felület folyamataira. A Clerk a felhasználóktól érkező kérések számára egy állandóüzenet-sorbanállási területet tart fenn (permanent message queue), az `InQ` osztály egyetlen példányát. Minden egyes eVote-folyamat saját ideiglenesüzenet-sorbanállási területtel rendelkezik, hogy fogadja a Clerktől érkező üzeneteket. Ezek az `OutQ` osztály példányai.

Osztott memória

Bár a szavazási eredmények és a felhasználó által kért statisztikai adatok cseréje az `OutQ` objektumokon keresztül zajlik, a lassan mozgatható adatokhoz egy további folyamatközi adatcsere-szol-

Az eszményi demokrácia

A demokrácia mind a vitát, mind a szavazást magában foglalja. Az elektronikus demokrácia általános modellje egy olyan webhely, ami választási lehetőségeket kínál, és megengedi a szavazást. Az effajta elektronikus demokrácia elleni legsúlyosabb kifogás, hogy ez politikaelméleti szemszögből nem is demokrácia. A kizárólag szavazásra korlátozódó demokráciát már 1982-ben bírálta *Jean Betheke Elshtain* politológus: „interaktív kagylójáték, ami abban a hitben ringat bennünket, hogy résztvevők vagyunk, holott valójában mindössze a külső ösztönző tényezőkből összetákolt rendszer válaszadó végét testesítjük meg”.
Elshtain azzal vádolja ezeket a rendszereket, hogy demokráciák helyett csupán népszavazások: „a népszavazási rendszerben a többség akarata elnyomja a kisebbség nézeteit vagy a kevésbé népszerű nézeteket. A népszavazás képes jól együttműködni a hatalomelvű politikával, amelyet a többségi nézetek álarca mögött visznek véghez, de legalábbis a kisebbségi nézetek feletti szemet hunyással. Ez a vélemény könnyen rögzíthető a játszi könnyedséggel manipulált, szertartásos népszavazásokkal, így az érdemi kérdések megvitatására nem is merül fel igény”. Egy másik politológus, *Brian Fay* így vélekedett a demokráciáról: „a polgárok részvétele a csoportos önazonosság-meghatározási folyamatban a legfontosabb”. Mindebből az szűrhető le, hogy a valódi demokráciában az elsődleges tevékenység a véleménycsere, nem pedig a szavazás. Egy tökéletes demokráciában lehetőség nyílna rá, hogy a kérdések terítékre kerüljenek, a véleménycsere megtörténjen, a vita lehetőségei között a saját álláspontunkkal megegyezőket felkutassuk, a kérdéseket újból megtárgyaljuk, és ismételt önellenőrzést folytassunk mindaddig, amíg közmegegyezés nem születik.
Az egyetértés azonban a demokráciának rendszerint nem a gyakorlati oldala, hanem az elméleti formája. A többségi szabály megosztja az adott csoportot, és fojtogatja a vitát. A közmegegyezés egyesíti a csoport tagjait, és megalapozott kölcsönös előnyökön alapuló döntésekhez vezet. A tökéletes demokráciában az új kérdések felvetésében minden résztvevő azonos joggal bír; egyenlő esélye van arra, hogy minden vitában részt vegyen; azonos lehetősége van rá, hogy minden döntésnél szavazzon és hogy a viták nyomán megváltoztassa szavazatát; továbbá minden szavazat egyenlő súllyal érvényesül minden egyes döntésben.
A tökéletes, közvetlen, közmegegyezésen alapuló demokrácia nem mindig gyakorlatias, még a számítógépes hálózaton sem lehet közvetlenül elérni. Számítógépes hálózataink azonban olyan eszközt kínálnak, amellyel talán meg lehet valósítani.

Forrás: Jean Betheke Elshtain Interactive TV – Democracy and the QUBE Tube (Interaktív televízió – demokrácia és a QUBE-cső?) 1982. aug. 7–14., The Nation, 108. oldal

gáltatás, az osztott memória használatos. A választási elemek listája és azok tulajdonságai az osztott memóriában kerülnek tárolásra, úgy, hogy valamennyi egyazon konferenciára bejelentkezett felhasználó láthassa őket. A választási lista elemeinek tulajdonságai között a következő jellemzőket találhatjuk meg:

- `public`, `private`, `if_voted` igen/nem, illetve számszerű szavazás;
- látható vagy láthatatlan, ahol a statisztikák a szavazás lezárása után válnak láthatóvá;
- egyéni vagy csoportos;
- a szavazás elnevezése.

A konferencia `ItemList` objektuma felelős az osztott memória karbantartásáért. Amikor a konferenciához új közvéleménykutatást írnak ki, és ha a választási elemek növekvő száma az osztott memória foltozását igényli, akkor az erről szóló üzenet valamennyi tevékeny eVote-ügyfélhez el lesz küldve. Ez a rugalmas értesítési szolgáltatás lehetővé teszi a felhasználók számára, hogy az „összejövetelt” valós időben rendezzék meg.

Adatállományok

A Clerk minden egyes konferencia vagy elektronikus levelezőlista számára három adatállományt tart fenn, továbbá egy általános adatállományt, ami az elektronikus levélcímeket és a hozzájuk kapcsolódó számszerű azonosítókat tartalmazza. A ***BallotBox*** (szavazó urna) a mintalista számára fenntartja a ***sample.dat*** nevű szavazócédulákat tartalmazó mintafájlt, a szavazó begépelt számszerű azonosítóját. Ezenkívül a ***BallotBox*** fenntartja a ***sample.bnf*** állományt, amelyben a ***sample.dat*** nyalábja (hash) és néhány hasznos statisztika található. A ***sample.inf*** állományban található a pillanatnyi elemek szavazólapokra leképezett rendszere, amit az `ItemList` objektum tart karban. Az eVoteban a felhasználói felület a folyamodványon szereplő aláírásokat és megjegyzéseket egy közönséges állományban tárolja, pontosabban egy folyamodványhoz egy állomány tartozik.

Felhasználói felületek

Ez idő szerint a Clerk két felhasználói felülettel bír, de több létrehozására ösztökél. A Clerk első felhasználói felülete egy egyszerű szövegalapú Telnet-felület, amit a BBS- és a konferenciarendszerekre gondolva terveztek.
Az Internet robbanásszerű elterjedése visszavetette a konferenciarendszerek iránti érdeklődést. Az elektronikus levelezés a kapcsolattartás meghatározó formájává vált, és a levelezőlisták a közösségi vita eszközeivé fejlődtek. Így az elektronikus levelezés felhasználói felületei olyan módon készültek, hogy azok a levelező közösségek számára formális döntések kialakítását tegyék lehetővé a még mindig nagy népszerűségnek örvendő Mailman levelezőlista-kezelő program felhasználásával.

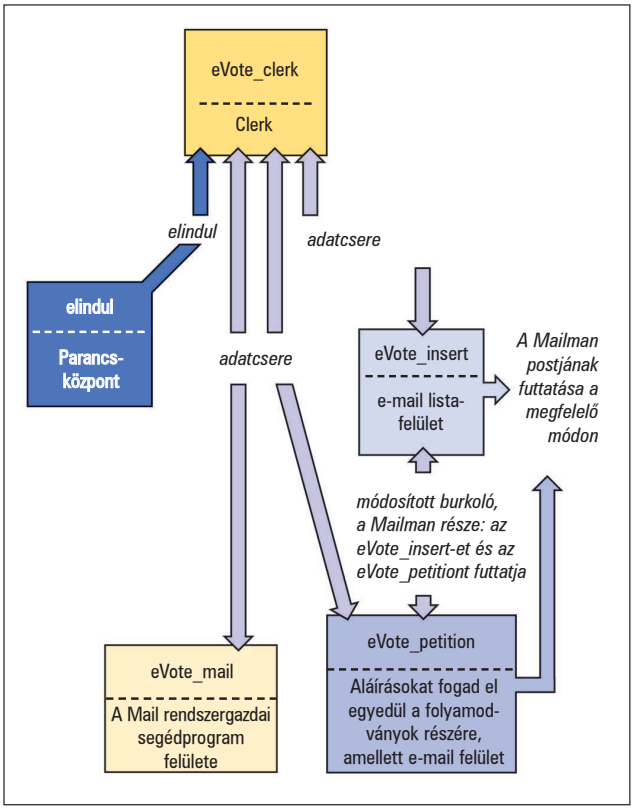
A levelezőfelület

Az eVote elektronikus levelezőlista felülete három részvételi szintet kínál:

1. Szavazó: a felhasználók leadhatják szavazataikat vagy meg is változtathatják őket; az adatok lekéréséhez jogosultságokkal bírnak, mint mondjuk a választás rendszergazdája vagy a lista tulajdonosa.
2. Felhasználó/rendszergazda: bármelyik felhasználó kezdeményezhet új választást. Általános körülmények mellett csak a szavazást kezdeményező felhasználó zárhatja le a választást, és távolíthatja el a listáról.
3. A lista tulajdonosa: néhány parancsjelszóval védett. Ezek felülbíráló érvényűek, úgyhogy a tulajdonos lezárhatja a választást, elvetheti a választási eredményeket, megváltoztathatja a szavazati előjogot, és a résztvevő szavazócéduláját új címre lehet költöztetni. A lista tulajdonosa az eVote használata nélkül megtarthatja kötelezettségeit és jogosultságait.

Öt végrehajtható állomány

Az elektronikus levelezés általános felépítése az *1. ábrán* látható. Az eVote tulajdonképpen öt együttműködő programot jelent: eVote_Clerk, the Clerk, eVote_insert, the e-mail list user interface (levelezőlista-felhasználói felület), eVote_mail és a mail administrator's utility interface (a mail rendszergazdai segédprogramjának felhasználói felülete), a Clerk vezérlőközpontja.



1. ábra Az eVote levelezőfelülete

Az eVote_Clerk folyamatosan fut a háttérben a számítógépen, új választásokat hoz létre; megszünteti a régi elemeket; elfogadja, sorba rendezi és tárolja a szavazatokat; valamint jelentést és kimutatást készít róluk. Az eVote_Clerk nem közvetlen felhasználói felület: az eVote végrehajtható fájl indítja el, szabályozza a működését és állítja le. Az eVote_insert végrehajtható állomány az a levelezőfelület, ami összehangolja a munkát a Mailmannel, a népszerű nyílt forrású levelezőlista-kiszolgálóprogrammal. A vitához szükséges közvetítőeszközt a Mailman „adja”, a szavazási szolgáltatást pedig az eVote. Ezt az együttműködést a levéltovábbító program (Mail Transfer Agent – MTA) alias állományában kell beállítani. Kivételt képez az az eset, amikor MTA-nak az Eximet választottuk és listakiszolgálónak a Mailmant. Az Exim beállítófájlja képes a listák kezelésére és az eVote-tal való együttműködésre.
Az eVote_mail lehetővé teszi a weblapot felügyelő rendszergazda számára, hogy a Clerk előfizetői listáját a Mailmanéhez igazítsa, vagyis azt összehangolja vele. A weblapot kezelő rendszergazda arra is felhasználhatja, hogy megakadályozza az

A választó lehetőségek tárháza: az eVote közvéleménykutatás-típusai

A választó a levelezőfelületen az igen/nem típusú szavazatleadáshoz az alábbi sort küldi el a lista címére. A közvélemény-kutatás során a szavazatösszesítés (vote tally) mindenki számára látható:

```
eVote poll [y/n] public visible
```

Az üzenet első szavának az eVote-nak kell lennie, különben a parancsot a teljes listának küldi el, és az eVote figyelmen kívül hagyja. Új közvélemény-kutatás kiírásához a `poll` parancs használandó. A szavazatok igen/nem típusúak vagy számszerűek lehetnek. Elsőként a szavazás korlátai szerepelnek szögletes zárójelek közé téve ([]). A legmagasabb adható szavazat 119, míg a legalacsonyabb -119, az [y/n] jelölések az igen, illetve a nem szavazatot jelölik. A `public`, azaz nyilvános szavazás a kézfeltartásos szavazást jelöli, a `private` a titkos szavazást, az `if_voted` segítségével pedig a résztvevők megtudhatják, hogy ki adta le a szavazatait – azt viszont nem, hogy ki milyen döntést hozott.

Csoportos szavazástípusok

A közösséget meg lehet kérni, hogy válasszon egyet a számos lehetőség közül. Az ilyen választást lehetővé tevő parancs a következő:

```
eVote poll group public visible 1
1. Félédes [Igen/Nem]
2. Félédes mandulával [Igen/Nem]
3. Tej [Igen/Nem]]
4. Tej mandulával [Igen/Nem]
```

Az üzenet a következő lesz:
Melyik csokoládét parancsolja?

A `group` kulcsszó azt jelzi, hogy több választási lehetőség is adott. A sor végén lévő egyes szám azt jelzi, hogy a választható elemek száma egyre van korlátozva. Ezután következik a választható elemek listája; végül ezek alatt találjuk az üzenet szövegét, a kulcsszót és a közvélemény-kutatáshoz kapcsolódó szöveget. Vegyünk egy kissé összetettebb példát:

```
1. épület-karbantartás [2,4]
2. rendszerkarbantartás [10,15]
3. személyzet [20,40]
4. fejlesztés [0,50]
5. távollét [0,50]
6. szórakozás [0,50]
```

Az üzenet az alábbi lesz:
Kérjük, működjön közre költségvetésünk kialakításában. A rendelkezésünkre álló pénzalapokat az összes szavazat arányában fogjuk elosztani. A jelenlegi közvélemény-kutatás arra kéri a résztvevőket, hogy száz szavazatot hat terület között osszanak el. Az egyes területeknél más-más alsó és felső határérték adható meg.

Amikor a lista valamelyik tagja elindít egy ilyen közvélemény-kutatást, valamennyi résztvevő megkapja a választási rendszer szabályait, mint az az alább bemutatott példán is látható:

```
Subject: [DEMOCRACY] Poll: Budget
On Mon, 11 Nov 2002 23:12:04 -0800
marilyn@rosita.deliberate.com attached a poll
to this subject:   Budget

==============================
A KÖZVÉLEMÉNY-KUTATÁS SZÖVEGE

Kérjük, segítsen elkészíteni a költségvetésünket!
A pénzalapokat az összes szavazat átlagához mérve fogjuk elosztani.

==============================
A KÖZVÉLEMÉNY-KUTATÁSSAL KAPCSOLATOS TUDNIVALÓK

6 résztvevőt kértünk meg, hogy 100 szavazatot az alábbi területek
között osszon el.

 Min.      Max.      Az Ön      Átlagos    A kívánt
szavazat  szavazat  szavazata  szavazat   terület

  2         4         –          –        1. épület-karbantartás
 10        15         –          –        2. rendszerkarbantartás
 20        40         –          –        3. személyzet
  0        50         –          –        4. fejlesztés
  0        50         –          –        5. túlköltekezés
  0        50         –          –        6. szórakozás

Az Ön rendelkezésére álló 100 pontból eddig 0-t jelölt meg.

======================
KÖZVÉLEMÉNY-KUTATÁS

Ez egy nyilvános közvélemény-kutatás, s ez azt jelenti,
hogy elektronikus levelet küldhet a következő címre:
        democracy@rosita.deliberate.com

A tárgynál tüntesse fel a Budget (költségvetés) szót, az üzenet
szövegében pedig:
        eVote who

parancsot annak érdekében, hogy az összes szavazatot tartalmazó
űrlapot megkaphassa.
Az alábbi parancs egy 140 oszlop széles táblázatot fog létrehozni. Az
előre beállított érték 80. Természetesen a demokráciálistán mindenki
más is figyelemmel kísérheti mások szavazatait.

==============================
HOGYAN ADHATJUK LE SZAVAZATAINKAT?

| 1. Küldjön elektronikus levelet a
|    democracy@rosita.deliberate.com címre.
|
| 2. A tárgynál tüntesse fel a Budget (költségvetés) szót.
|    (Ne törődjön a válasz által (reply to) a tárgy mezőben
|    létrehozott többi szóval.)
|
|    ************************
----> * Fontos: ez a két lépés könnyű.          *
      *          Ennél az üzenetnél használja   *
      *          a reply-to (válasz) gombot     *
      ************************
```

3. Az üzenetnek az eVote szóval kell kezdődnie, különben a levél a teljes demokrácia-levelezőlistához kerül, és be sem lesz számítva. Amennyiben a második elemnél mondjuk 12-t, illetve a 6. elemnél 25-öt szeretnénk beállítani, ezt az alábbi üzenettel tehetjük meg:

```
eVote
2. 12
6. 25
```

A választható elemek listája olyan hosszú lehet, amennyire csak szükséges. Minden olyan elem, amelyik nem kapott szavazatot, az adott elemhez tartozó választási lehetőségnél a lehető legkisebb értékre lesz beállítva.

4. Ha az üzenetedben nincs aláírás, illetőleg bármilyen egyéb szöveg található a szavazólap alatt, akkor a szavazatot követően írjuk be az end szócskát.

==========

A LEADOTT SZAVAZAT MEGVÁLTOZTATÁSA

A szavazóprogram révén mindaddig meg lehet változtatni a leadott szavazatokat, amíg a közvélemény-kutatás zajlik.

==========

A SZAVAZAT TÖRLÉSE

A költségvetésre (Budget) leadott szavazatok törléséhez a következő parancsot használhatjuk:

```
send the message:
      eVote remove
```

==========

A VÁLASZTÁSI EREDMÉNYEK MEGTEKINTÉSE

A költségvetésre leadott, és már összeválogatott szavazatok megtekintéséhez ezt a parancsot kell használnunk:

```
      eVote info
```

==========

RÉSZLETES TÁJÉKOZTATÁS

A költségvetésről (Budget) bővebb tájékoztatás kapható:

1. Küldj egy elektronikus levelet az alábbi címre:
   democracy@rosita.deliberate.com
2. A tárgyhoz írjuk be:
   Budget
3. Ha a `who` parancsról szeretnénk részletesebb adatokhoz jutni:

```
      eVote help who
```

Ha az üzenet így szól:
eVote how Charlie@somewhere.com

Erre válaszul meg fogod kapni a Charlie szavazatairól szóló jelentést. A saját választási eredményeidet és a pillanatnyi szavazatok összesítését a következő paranccsal lehet elérni:

```
      eVote info
```

A Vote/Mailman programról – bármilyen tárgy megnevezésével – általános tájékoztatást kaphatunk, és máris küldhetjük az üzenetet.

```
      eVote help
Clerk 2.520 Ser.No. 1
```

A felhasználó az alábbi üzenet küldésével felelhet:

```
Subject: Re: [DEMOCRACY] Poll: Budget
1. 3
2. 12
3. 30
4. 25
5. 25
6. 5
```

Az eVote elektronikus nyugta küldésével válaszol. A nyugtában szerepel a listatag eredeti szavazási üzenete az összes fejléccel együtt.
Ha a felhasználó valamilyen hibát ejt, az eVote hibaüzenettel próbál segíteni: *6 résztvevő és 5 szavazó!*

Végül álljon itt az `eVote who` által előállított táblázat, amiből megtudhatjuk, hogy ki hogyan szavazott.

```
    ,------------------------------------ 1 building maintenance
    |     ,------------------------------ 2 system maintenance
    |     |     ,------------------------ 3 staff
    |     |     |     ,------------------ 4 development
    |     |     |     |     ,---------- 5 outreach
    |     |     |     |     |     ,-- 6 party
    1     2     3     4     5     6
 ------ ------ ------ ------ ------ ------
|    3 |   12 |   30 |   25 |   25 |    5 | marilyn@rosita.deliberate.com
|    4 |   11 |   32 |   23 |   18 |   12 | barry@rosita.deliberate.com
|    3 |   11 |   33 |   21 |   19 |   13 | judi@rosita.deliberate.com
|    4 |   14 |   26 |   26 |   15 |   15 | charlie@rosita.deliberate.com
|    4 |   13 |   29 |   26 |   19 |    9 | jane@rosita.deliberate.com
|      |      |      |      |      |      | clint@rosita.deliberate.com
 ------ ------ ------ ------ ------ ------
|    2 |   10 |   20 |    0 |    0 |    0 |    MINIMUM VOTE
 ------ ------ ------ ------ ------ ------
|    4 |   15 |   40 |   50 |   50 |   50 |    MAXIMUM VOTE
 ------ ------ ------ ------ ------ ------
|  3.6 | 12.2 | 30.0 | 24.2 | 19.2 | 10.8 |    AVERAGE
```

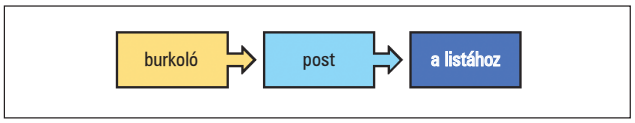
Folyamodványtípusok (Petition Types)

Bármelyik típus mellékelhető egy folyamodványhoz, ami lehetővé teszi a részvételt olyan emberek számára, akik nincsenek rajta az elektronikus levelezőlistán. A folyamodványon az egyes elemekhez megjegyzéseket lehet fűzni.

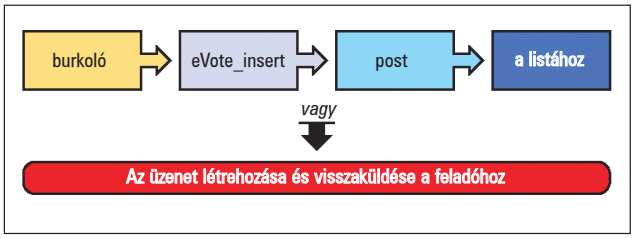
Jövendő közvéleménykutatás-típusok

A Clerk olyan felépítésű, ami más közvéleménykutatás-típusok beillesztését is lehetővé teszi.
Néhány érdekes lehetőségről a
➲ http://www.michcom/~donald/dispute.html és a
➲ http://accuratedemocracy.com/a_map.htm címeken olvashatunk.

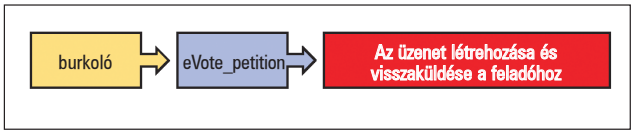
egy adott címről történő szavazatok leadását, vagy egy címet az összes listáról eltávolítson. Ugyanakkor e program a megerősítésre való várakozás közben elavulttá vált üzenetek törlésére is alkalmazható.
Az eVote levelezőfelületén két szolgáltatás érhető el: a levelezőlisták lekérdezése és a folyamodványok kezelése. A folyamodványkezelő szolgáltatás a részvételt mindenki számára megengedi, ezzel szemben a levelezőlista-kezelő szolgáltatás kizárólag levelezőlisták részvételét engedélyezi. A folyamodványok kezelése a folyamodványlisták tagjainak kezelésével közösen zajlik, vagyis olyan tetszőleges listák kezelésével, amelyek a petition szóval kezdődnek: petitiona, petitionb stb. A folyamodványlistákban indított lekérdezéseket azok is megnyithatják, akik az adott listában nem tagok.



2. ábra A Mailman listakezelő az eVote nélkül



3. ábra Az eVote ellenőrzi a postát



4. ábra Folyamodványszolgáltatás

Az eVote végrehajtható állománya voltaképpen az egész eVote programcsomag parancsközpontja, ami különféle paraméterekkel indítható el. A megadott kapcsolóktól függően az eVote megkezdi vagy éppen befejezi működését, megvizsgálja a Clerket, ellenőrzi és összesíti (összehangolja) az adatokat, valamint törli a naplót és újraindítja a naplózást.
A Mailman bármilyen tetszőleges levéltovábbító program – úgymint Sendmail, Exim vagy a Postfix – által meghívható. Ha minden a szokványos módon zajlik, akkor a levelezőlistacímre irányított küldemény csővezetéken keresztül a Mailman burkolójához (wrapper) kerül a folyamatra vonatkozó engedélyek ellenőrzése végett, másrészt a csővezetékben végrehajtott programok korlátozása érdekében. Ezt követően a Mailman meghívja a post-héjat, hogy az szétszórja a lista címeire a leveleket.
Az egyszerű Mailman-listához tartozó ***sample*** (minta) nevű aliasbejegyzés az alábbihoz hasonló lesz:

```
sample:              "|/home/mailman/bin/
                       wrapper post sample"
sample-admin:        "|/home/mailman/bin/wrapper
                       mailowner sample"
sample-request:      "|/home/mailman/bin/wrapper
                        mailcmd sample"
sample-owner:        sample-admin
```

A `mailcmd` programnak szüksége van néhány sornyi programkódra, ami arra utasítja, hogy elektronikus levélben értesítést küldjön az eVote-nak, valahányszor valaki feliratkozik egy listára vagy éppen lemond egy ilyen „előfizetést”.
A listatagokhoz csoportos üzenetszórással eljuttatandó üzenet a sample alias révén a Mailman postázó (post) programjához kerül. Az eVoting bekapcsolásához az `eVote_insert` parancsot be kell szúrni a csővezetékbe.

```
sample: "|/home/mailman/bin/wrapper
          eVote_insert post sample"
```

A burkoló C-forráskódja néhány módosítást fog kapni, hogy az `eVote_insert` modul futtatását lehetővé tegye.

Most érkezik el a pillanat, amikor az eVote először ellenőrzi a lista csoportos üzenetszórási címére érkező küldeményeket. Ha a bejövő üzenet első szava `eVote`, akkor az eVote a küldeményt szavazatfeldolgozás végett máris eltéríti. Különben az üzenetet továbbküldi a postázóhoz (post) (lásd a *3. ábrán*).
A folyamodványlisták telepítése pontosan olyan, mint a többi eVote-listáé. Mint azt már korábban említettük, a program felismeri ezek különleges jellegét, mert ezek nevében is szerepel a `petition` szó. Ezek az állományok szolgálják a folyamodványok felügyeletében való közreműködést. Egy folyamodványlista tagjai vitába szállhatnak egymással, lekérdezhetik az adataikat, és olyan hatalommal rendelkeznek, hogy az egész világra kiterjedő választást is kiírhatnak. Ezek a folyamodványok bármilyen tetszőleges eVote-szavazattípust tartalmazhatnak, és az aláíróknak rendszerint megjegyzéseket is kell fűzniük a szavazatukhoz.
A folyamodványszolgáltatás az aláírások fogadására egy további aliasszal is rendelkezik:

```
eVote: "|/home/mailman/mail/wrapper
         eVote_petition"
```

Az egyetlen `eVote_petition` alias a szolgáltatáshoz tartozó összes folyamodványlista aláírását feldolgozza.

Két kiegészítés a választásokhoz

Jelenlegi formájában az eVote/Clerk rendszert nem az elnökválasztások lefolytatására tervezték, és nem is alkalmas rá. Két jelentősebb kiegészítéssel azonban a legbiztonságosabb és legpontosabb megoldássá válhat.

1. ***A hálózati réteg:*** használata révén a hálózatba kapcsolt Clerkek végzik az osztott adatok kezelését. Ugyancsak a hálózat segítené elő más Clerkek adatainak és számításainak ellenőrzését. Vegyük például azt az esetet, amikor a szavazó az egyik Clerknél szavaz, és a bizonylat az adott Clerktől elkerül, majd később a szavazó egy másik bizonylatot kap egy másik, véletlenül kiválasztott Clerktől – ez a folyamat biztosítaná valamennyi Clerk egységességét (integrity).
2. ***A biztonságos titkosítási réteg:*** így csak a program és a szavazó láthatja a szavazó választását. Ezekkel a kiegészítésekkel a Clerk képes lenne az általános célú adatbázis-kiszolgálóknál nagyobb biztonságot nyújtani, mivel a Clerk átfedéssel és a földrajzi távolságokat áthidalva képes a szavazólapokat eljuttatni a kis gépekből álló, GNU-rendszert futtató hálózatokra. Ezenkívül azt is fontos megemlíteni, hogy egy-egy rendszergazdának jelentéktelen kötelezettsége és hatalma van a rendszer felett, mivelhogy minden rendszergazdát a Clerkek hálózata tart „szemmel”. Végül a Clerk magában foglalja a szavazókat is, hiszen azok nemcsak újraszámlálásokat, többször ismétlődő ellenőrzést jelentenek, hanem esetenként újraszavazást is. Egy ilyen rendszerrel magabiztosan vághatunk neki az elektrodemokratikus kornak.

Linux Journal 2003. március, 107. szám

Marilyn Davis (marilyn@deliberate.com)
A régmúltban szerzett PhD fokozatot az elméleti rádiócsillagászat területén. Jelenleg Python, C nyelveket és GNU fejlesztőeszközöket tanít, de felszolgálással is foglalkozik a kaliforniai Sunnyvale-ben, az UCSC levelezőoktatásban.

Faxkiszolgáló a fax2senddel

A legismertebb szabad faxkiszolgáló, a Hylafax mellett érdemes megismerkedni a „fizetős” fax2send kiszolgálóval is.

A számítógépes faxkiszolgálók alkalmazása nemcsak a papírmentes iroda, de a faxok kényelmesebb kezelése miatt is egyre növekvő igény a kisvállalkozások, irodák körében. Ha már linuxos fájl-, illetve nyomtatókiszolgáló üzemel az irodánkban, érdemes a gépet egy újabb feladattal is megbízni. Nézzünk meg egy nem szabad, de nagyon egyszerűen beállítható és könnyen, megbízhatóan használható megoldást!
A fax2send két részből áll: egyrészt a kiszolgálóprogram, másrészt a windowsos ügyfélprogram részből. A linuxos X-es ügyfelek jelenleg fejlesztés alatt állnak. A kiszolgálóprogrammal együtt letölthető X-es ügyfelek egyelőre csak azon a gépen működnek, amin a kiszolgáló is megtalálható.
A programokhoz könnyen hozzájuthatunk a ➲ http://www.fax2send.com weboldalról. Mi most az „Installation wizard” változattal foglalkozunk, mert a gép Debian linuxos.
A letöltött tömörített fájlt kicsomagoljuk, majd egyszerűen elindítjuk az `install` parancsfájlt. A telepítés néhány kérdés megválaszolásából áll. Ha a telepítést X alól indítjuk, akkor grafikusan zajlik, de nyugodt szívvel feltételezhetjük, hogy a kiszolgálókon nincs igazán szükség grafikus felületre. Először rögtön elolvashatjuk a felhasználás feltételeit, majd miután azt elfogadtuk, újabb kérdés vár minket, miszerint indulhat-e a telepítés.
A telepítés 15–20 másodperc alatt lezajlik, ezt követően újabb kérdést kapunk. El kell döntenünk, hogy próbaváltozatként vagy bejegyzett változatként szeretnénk-e használni a programot. Válasszuk a próbaváltozatot, amivel a faxkiszolgálót teljes szolgáltatáskörrel egy hónapig lehet használni. Ezt követően a modem beállításával folytatódik a telepítés menete. Kiválaszthatjuk, hogy modemünk melyik kapuhoz csatlakozzon, majd a program le is ellenőrzi a csatlakoztatott modemet.
Itt álljunk meg egy szóra. A fax2send akkor fog megfelelően működni, ha a modemünk ismeri a Class 2 vagy 2.0 protokollt. Ezt a tesztprogram megvizsgálja és közli az eredményt. Ha a modemünk megfelelt, már csak arra kell választ adnunk, hogy milyen lapformátumot szeretnénk használni. Választhatunk az A4, illetve a Letter formátum között. Ezzel a faxkiszolgáló telepítése véget ért – nem volt bonyolult folyamatnak nevezhető. A telepítés után a fax el is indul és közli, hogy ne feledkezzünk meg a beállításokról. Elsősorban a felhasználói jogosultságokat adjuk meg, amiket a */etc/fax2send/userlist* fájlban állíthatunk be. Ha például a munkaállomás neve ***ws1***, akkor a felhasznalo@ws1-et adjuk meg a ***userlist***-ben. A faxkiszolgáló újraindítása után már ezzel a felhasználóval jelentkezhetünk be, feltéve, hogy az ügyféloldali programot is telepítettük.
Azért még ne szaladjunk előre! A kiszolgálóoldalon érdemes még két beállítást elvégezni. Elsőnek a nyomtatónkat – ha van – vegyük fel a */etc/fax2send/printers* fájlba. Ez jó lesz majd, ha azt szeretnénk, hogy a beérkezett faxok egyből kinyomtatásra kerüljenek. A másik, papírtakarékosabb módszer – ha levélkiszolgáló is akad a gépünkön –, hogy az érkezett faxokat egy elektronikus postafiókba irányítjuk át. Ehhez a */etc/fax2send/routing* fájlt kell egy kicsit módosítanunk. Ha a *routing* fájlba beillesztjük az alábbi sort, akkor az érkezett fax azonnal nyomtatásra kerül, és a tamas nevű felhasználó levélben is megkapja a faxot, feltételezve, hogy a nyomtatót „hp”-ként adtuk meg.

```
default~~~hp~tamas~Unsorted
```

Így már teljesen készek állunk faxok fogadására.
Most nézzük meg, hogyan tudunk faxot küldeni, illetve érkezett faxokat kezelni a windwosos munkaállomásokon, ha ezt

nem (csak) a levelezőprogramunkban szeretnénk megtenni. Az ügyfélprogram a fax2send weboldaláról ugyancsak letölthető, mindegyik Windows-változathoz használható. Telepítése szintén nem igényel pilótavizsgát, a szokásos Windows-telepítőkhöz hasonlóan kell eljárnunk. Sok teendőnk nincs, szinte csak a ***next*** gombot kell nyomogatnunk és a szokásos kérdésekre kell válaszolnunk, hogy hová és milyen néven szeretnénk telepíteni a programot. Első indításkor meg kell adnunk a felhasználót, aki kapcsolódni szeretne a kiszolgálóhoz. Ez egyezzen meg azzal, akit a ***userlist*** fájlban beállítottunk. Ezután már készen állunk a faxok kezelésére és a faxküldésre.
A nyomtatók közé bekerült egy fax2send nyomtató. Ennek segítségével tudunk faxokat küldeni. Érdemes a fax2send programot az indítópultba betenni, így a rendszer indulásakor önműködően elindul. Ha ezzel a nyomtatóval nyomtatunk, akkor bekéri a faxszámot, ahová a faxot küldeni szeretnénk, továbbá eldönthetjük, hogy fedlappal vagy fedlap nélkül továbbítjuk-e. Ha elküldjük a faxot, hamarosan hallhatjuk, hogy a modem tárcsázni fog, és az üzenetet megpróbálja eljuttatni a címzettnek.

Ambrits Tamás (ambrits@ambrits.hu)
Nem „fanatikus” Linux-rajongó, de egyre több feladat megoldására alkalmazza nagy megelégedéssel.

A Firewall Builder használata (2. rész)

A bástyagép beállítása és az IP Tables tűzfalszabályainak megadása után már tisztán láthatjuk a biztonsági házirend lényegét.

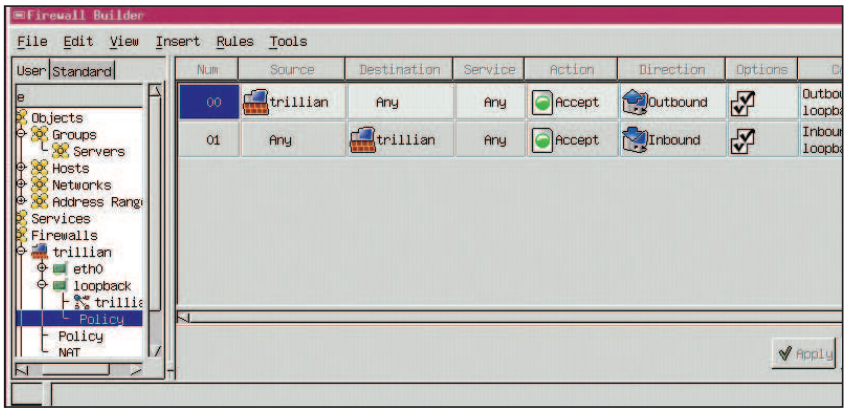
A múlt alkalommal a Firewall Builderben elkészítettük azokat az újrafelhasználható objektumokat, amelyeket az IP Tables-szabályokban fogunk alkalmazni. Ez alkalommal két szabálykészletet fogunk a Firewall Builder segítségével létrehozni: az egyik a bástyagép számára készül el, amelynek önmagát kell megvédenie, a másik egy tűzfal számára, amelynek egy teljes hálózat biztonságáról kell gondoskodnia.

Helyi szabályok a bástyagépen

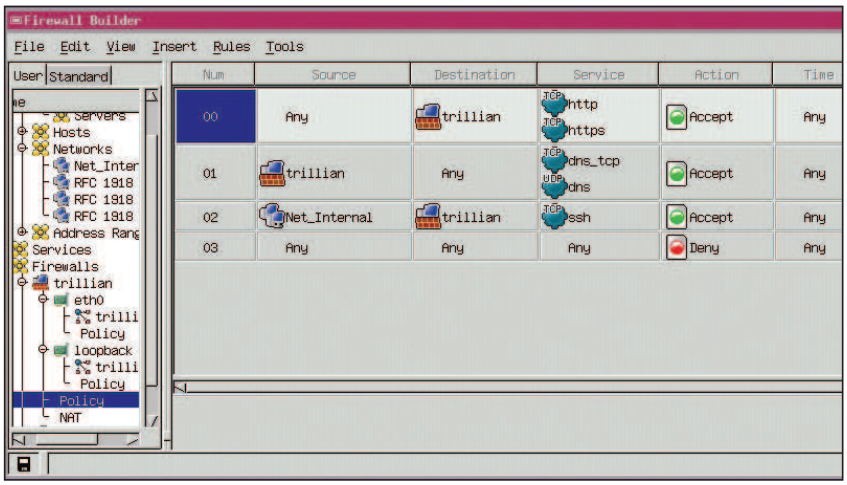
Elsőként tekintsük át a bástyagép helyzetét. A Netfilter/IP Tables, illetve általában a csomagszűrés kapcsán széles körben elterjedt tévhit, hogy a csomagszűrés kizárólag a tűzfalak feladata. Ha valóban biztonságos rendszert szeretnél, elég balga ötlet volna a biztonságot egyetlen elemre bízni. Természetesen szükség van egy gondosan beállított és felügyelt tűzfalra, ami az összes hálózatra csatlakozó számítógép számára biztosítja a védelmet, ám a számítógépek maguk is képesek kell legyenek önmaguk megvédésére – különösen a bástyagép, amelyen nyilvánosan elérhető szolgáltatásokat (ftp, www) futtatunk. Tegyük fel például, hogy nyilvános webkiszolgálódon 2.4-es Linux fut. A gépen helyi Netfilter-szabályok segítségével külön védelmi réteget kell emelni arra az esetre, ha egy támadó megtörné a vállalati tűzfalat, vagy egyéb módon jutna keresztül rajta. Ha a kiszolgálón 2.4-esnél korábbi rendszermag fut, akkor a Netfilter/IP Tables helyett IP Chainst kell használni. Ebben az esetben külső fejlesztőktől szerezhetsz IP Tables-IP Chains fordítómodult, és ennek segítségével használhatod a Firewall Buildert a parancsfájlok elkészítésére.

Visszacsatoló szabályok

Minden tűzfal beállításakor az első lépés – így a bástyagép esetében is – az, hogy teljhatalmat adunk a helyi visszacsatoló felületnek. A visszacsatolóra számos, a helyi folyamatok és démonok között zajló átvitel során van szükség. A helyi hurkot engedélyező szabályok hiányában az IP Tables parancsfájl futtatásakor olyan szolgáltatások szakadnak meg, mint a névszolgáltatás gyorstárazása vagy az SSH-kaputovábbítás.



1. kép A visszacsatoló felületre vonatkozó szabályok

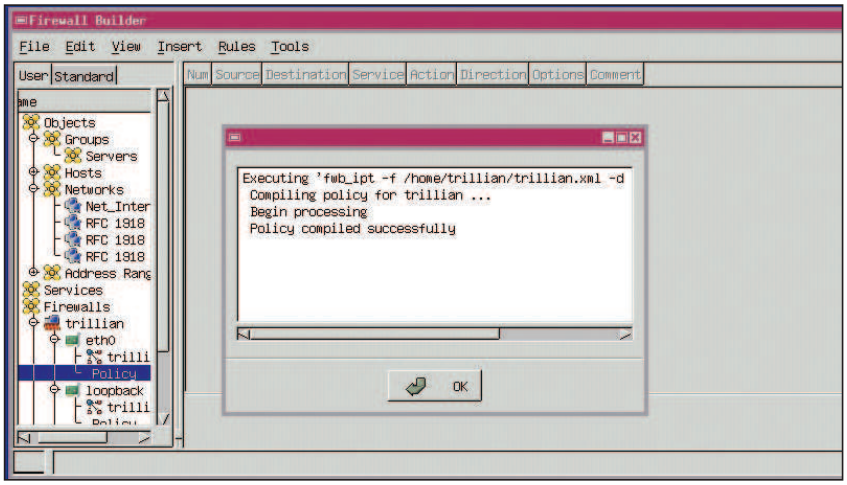


2. kép A bástyagép házirendje

Tegyük fel, hogy van egy Trillian nevű webkiszolgálód, ezt szeretnéd védeni. A Firewall Buildert feltelepítetted rendszergazdai munkaállomásodra – gondolom, még nem felejtetted el, hogy az X Window System, illetve az X alapú alkalmazások nem a bástyagépre valók. Létrehoztad azokat az objektumokat, amelyek a környezetedben lévő állomások, hálózatok és csoportok leírására alkalmasak, továbbá egy Trillian tűzfalobjektumot, majd mindezt egy helyi visszacsatoló felülettel egészítetted ki. Másként fogalmazva: elvégezted a múlt alkalommal ismertetett tennivalókat.

A Trillian helyi visszacsatoló felületéhez két szabályra lesz szükség: az egyik a kimenő, a másik a bejövő forgalmat engedélyezi rajta. A két szabály létrehozásához a következő lépéseket kell követned (*1. kép*):

1. A tűzfal visszacsatoló felületének alobjektuma alatt jobbra, a Firewall Builder ablakának bal oldali részén (az 1. képen loopback névvel szerepel) válaszd ki a visszacsatoló felület házirendjét – egyelőre üresnek kell lennie.



3. kép A házirend lefordítása

2. Kattints a ***Rules*** (Szabályok) menü ***Append rule*** (Szabály hozzáfűzése) pontjára, ekkor az ablak jobb oldali részében egy üres szabály jelenik meg.
3. Húzd a Trillian név melletti tűzfalikont az üres szabály ***Source*** (Forrás) mezőjébe. Ügyelj arra, hogy a kurzornak pluszjellé (+) kell változnia, mielőtt az egérgombot felengednéd.
4. Kattints az egér jobb oldali gombjával az új szabály ***Action*** (Művelet) mezőjébe, majd a menüből válaszd az ***Accept*** (Elfogadás) pontot.
5. Kattints az egér jobb oldali gombjával a szabály ***Direction*** (Irány) mezőjébe, majd válaszd az ***Outbound*** (Kimenő) pontot.
6. Kattints az egér jobb oldali gombjával a szabály ***Options*** (Beállítások) mezőjében található papír- és ceruzaikonra, majd válaszd a Turn logging ***OFF*** (Naplózás kikapcsolása) beállítást.
7. A jobb oldali egérgombbal kattints újra a szabály ***Options*** mezőjébe, majd válaszd a ***Modify options*** (Beállítások módosítása) parancsot. A megnyíló ablakban jelöld be az alul található jelölőnégyzetet, ezzel kikapcsolod az állapotalapú vizsgálatot. Semmi szükség arra, hogy a processzor erőforrásait a helyi hurkon keresztül folyó forgalom állapotalapú követésére pocsékoljuk.
8. Ha gondolod, az egér jobb oldali gombjával kattints az új szabály ***Comment*** (Megjegyzés) mezőjébe, válaszd az ***Edit Comment*** (Megjegyzés szerkesztése) parancsot, és készíts rövid feljegyzést a szabály szerepéről, céljáról (például „Helyi visszacsatoló felületkimenet engedélyezése”).

Az 1. képen is látható második szabály létrehozásához ismételd meg a 2–8. számú lépést. A 3. lépésben a Trillian ikonját értelemszerűen az új szabály ***Destination*** (Cél) mezőjébe kell húznod, és nem a forrásba. Az 5. lépésben az irányt ***Inbound*** (Bejövő) értékre állítsd.
Jogos a kérdés: pontosan hogyan működnek ezek a szabályok? Először is hangsúlyoznom kell, hogy kizárólag a helyi visszacsatoló felületre vonatkoznak. Bármelyik felületnek lehetnek saját szabályai, ezek feldolgozása a tűzfal általános szabályainak érvényre juttatása előtt történik meg. Bár ennél a két visszacsatoló szabálynál a Trillian cél és forrás is volt egyben, szó sincs arról, hogy e két szabály a Trillian bármelyik IP-címére is érvényes lenne. Kizárólag a helyi visszacsatoló felületen kimenő és beérkező csomagokra vonatkoznak.
El is érkeztünk a visszacsatoló szabályokkal kapcsolatos utolsó tudnivalóhoz. Feleslegesnek tűnhet két olyan szabályt létrehozni, amelyek ugyanúgy a tűzfalra vonatkoznak, és nem egyetlen, bármely forrást és bármely célt elfogadó szabályba sűríteni őket. Jómagam azonban úgy tapasztaltam, hogy amikor egyetlen szabályt hoztam létre, a Firewall Builder az `INPUT` és az `OUTPUT` helyett a `FORWARD` láncba írta a visszacsatoló szabályokat, ami viszont a visszacsatoló felület használhatatlanná válását okozta. Ha külön kimeneti és bemeneti szabályt hoztam létre a visszacsatolóhoz, a hiba eltűnt. Aggodalomra nincs ok, tapasztalataim szerint ez az egyetlen olyan helyzet, amikor a Firewall Builder rossz láncba írja a szabályokat. A hiba csak egylaki gépeknél jelentkezett, több felülettel is rendelkező tűzfalaknál nem.

A bástyagép házirendje

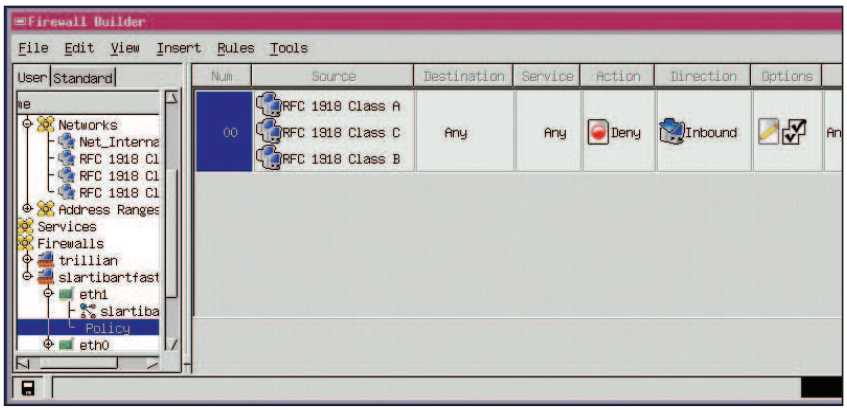
Miután a bástyagép visszacsatoló felületét lerendeztük, fordítsuk figyelmünket az általános házirendre. Ehhez némi tervezésre is szükség lesz. A cél nyilván az, hogy a Netfilter megfelelő védelmet nyújtson, de eközben a használhatóságot se korlátozza. Példagépünk, a Trillian egy webkiszolgáló, a többi állomás tehát HTTP és HTTPS protokollon keresztül fogja elérni. A megfelelő naplózás érdekében szeretnénk, ha a Trillian DNS-lekérdezéseket is végre tudna hajtani. Emellett bizonyos felügyeleti jellegű kapcsolatokra is szükség lehet. Erre a célra SSH-t fogunk használni, tehát a bejövő SSH-kapcsolatokat is engedélyeznünk kell, de csak a belső hálózat felől. A *2. képen* egy ilyen, a Trillian számára készített házirendet láthatunk.
Az egyes szabályok létrehozásának aprólékos ismertetésétől most eltekintenék, néhány dologra viszont érdemes kitérni. Először is az ablak bal oldalán megjelenő objektumfában látható, hogy közvetlenül a Trillian alatti ágban választottam ki az általános ***Policy*** (Házirend) objektumot, és nem a felületekre külön vonatkozó házirendobjektumok között. Nemcsak a Trillian objektumot használtam fel, hanem egy ***Net_Internal*** nevű hálózatobjektumot is, ami a múltkori írásom egyik példaobjektuma. Ez egy teljes hálózatnyi IP-címtartományra hivatkozik, pontosabban a 192.168.111.0 hálózatra. Míg a második szabály forrásként egyetlen IP-címet jelöl meg (a Trillian IP-címét), addig a harmadik szabály mindazokra a csomagokra érvényes, amelyeknek a forrás IP-címe a 192.168.111.1 – 192.168.111.254 tartományba esik.
Egy másik fontos tanács a szabályok létrehozásával kapcsolatban: bátran használd fel a Firewall Builder beépített szolgáltatásobjektumait, amelyeket a bal oldali ablakrész ***Standard*** fülére kattintva érhetsz el. Némi figyelem természetesen nem árt, mert ha a szolgáltatásobjektumok (például `dns_tcp`) szabályokba való húzogatásától eltérő műveletet is végzel, akkor az ablak jobb oldalán látható szabályok helyett az éppen kiválasztott elem tulajdonságai jelennek meg.
Egy kicsit pontosítva, ha éppen egy házirendet állítasz össze, akkor a ***Standard*** fülre, majd az objektumfa + (lenyitás) és – (összezárás) ikonjaira kattintva, illetve a szolgáltatásobjektumokat a megfelelő helyre húzva kényelmesen dolgozhatsz, a jobb oldali ablakrész által

A trillian.fw fájl tartalma

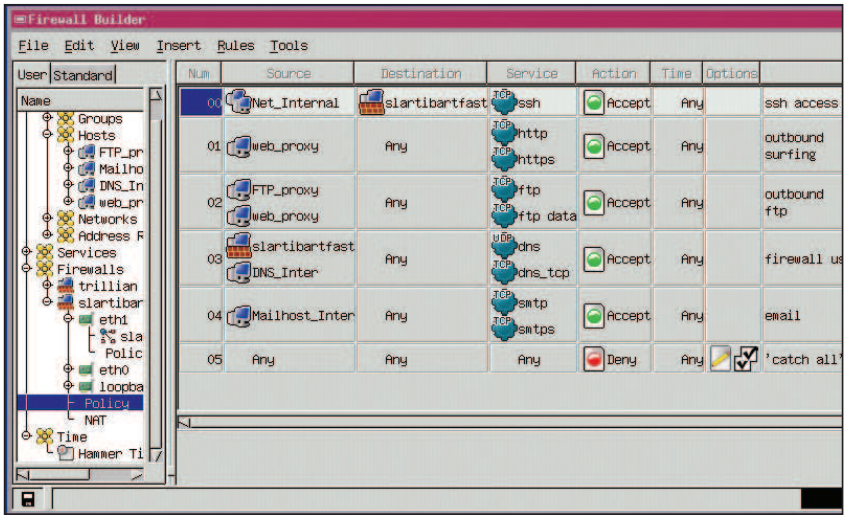
```
#!/bin/sh
#
#  Firewall Builder  fwb_ipt v1.0.8-3
#
#  Generated Tue Mar 11 08:01:21 2003 CST

log() {   if test -x "$LOGGER"; then
    logger -p info "$1"
  fi
}

MODPROBE="/sbin/modprobe"
IPTABLES="/sbin/iptables"
IP="/sbin/ip"
LOGGER="/usr/bin/logger"

cd /etc || exit 1

log "Tűzfalparancsfájl indítása…"

va_num=1

FWD=´cat /proc/sys/net/ipv4/ip_forward´
echo "0" > /proc/sys/net/ipv4/ip_forward
echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
echo 1800
➥> /proc/sys/net/ipv4/tcp_keepalive_intvl

$IP -4 neigh flush dev eth0
$IP -4 addr flush dev eth0 label "eth0:FWB*"

$IPTABLES -P OUTPUT  DROP
$IPTABLES -P INPUT   DROP
$IPTABLES -P FORWARD DROP

cat /proc/net/ip_tables_names | while read
     ➥table; do
  $IPTABLES -t $table -L -n |   while read c
              ➥chain rest; do
      if test "X$c" = "XChain" ; then
        $IPTABLES -t $table -F $chain
      fi
  done
  $IPTABLES -t $table -X
done

$IPTABLES -A INPUT   -m state --state
➥ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT  -m state --state
➥ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -m state --state
➥ESTABLISHED,RELATED -j ACCEPT

# Rule 0(lo):  Outbound from loopback
$IPTABLES -A OUTPUT  -o lo -j ACCEPT

# Rule 1(lo): Inbound from loopback
$IPTABLES -A INPUT  -i lo -j ACCEPT

# Rule 0(global)
$IPTABLES -A INPUT -p tcp  -m multiport
➥--destination-port 80,443  -m state
➥--state NEW -j ACCEPT

# Rule 1(global)
$IPTABLES -A OUTPUT -p tcp  --destination-
➥port 53 -m state --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p udp  --destination-
➥port 53 -m state --state NEW -j ACCEPT

# Rule 2(global)
$IPTABLES -A INPUT -p tcp  -s
➥192.168.111.0/24  --destination-port 22
➥-m state --state NEW -j ACCEPT

# Rule 3(global)
$IPTABLES -N RULE_3
$IPTABLES -A FORWARD -j RULE_3
$IPTABLES -A RULE_3 -j LOG  --log-level
➥warning --log-prefix "RULE 3 -- DROP "
➥--log-ip-options
$IPTABLES -A RULE_3 -j DROP

echo 0 > /proc/sys/net/ipv4/ip_forward
```

megjelenített adatok köre nem változik meg. Ha viszont egyszerűen kiválasztasz egy szolgáltatásobjektumot vagy -kategóriát a ***Standard*** fában (olyan módon, hogy rákattintasz, de nem húzod át sehova), akkor ennek az objektumnak a tulajdonságai jelennek meg a jobb oldalon. Ilyenkor vissza kell lépned a ***User*** (Felhasználó) lapra, újra ki kell választanod a tűzfal általános házirendjét, és csak ekkor jelennek meg újra a szabályok. Adatvesztés nem történik, de kényelmetlen és bosszantó jelenség. Sokkal fontosabb, hogy ezeknél a szabályoknál az állapotalapú vizsgálatot bekapcsolva hagytam, vagyis kihagytam a visszacsatoló szabályok kapcsán ismertetett eljárás 7. lépését. Normál esetben szeretnénk, ha a rendszermag megőrizné a hálózati átvitelekkel kapcsolatos állapotadatokat – éppen ez az, aminek köszönhetően a legtöbb átvitel egyetlen kétirányú szabállyal leírható, és nincs szükség két-két egyirányú szabályra. Az állapotalapú vizsgálatnak köszönhető például az, hogy ha egy átvitel megfelel a *2. képen* látható 2. szabály előírásának, ami engedélyezi a bejövő SSH-forgalmat a belső hálózat gépeiről, akkor a Trillian rendszermagja nemcsak a bejövő SSH-csomagokat engedélyezi, de a Trillian által válaszként küldötteket is. Ha a második szabálynál kikapcsoltam volna az állapotalapú vizsgálatot, akkor a Trillian 22-es TCP-kapujáról származó csomagok továbbításának engedélyezésére, és ezzel a válaszok elküldésének lehetővé tételére egy másik szabályt is létre kellett volna hoznom.

Végül az utolsó kivételével mindegyik szabály esetében ki kell kapcsolni a naplózást, a fenti eljárás 6. lépésében leírt módon. Sokan úgy gondolják, nincs értelme és haszna lemezhelyet és be-, illetve kiviteli terhelést áldozni arra, hogy a rendszer a tűzfalszabályok által feldolgozott csomagok mindegyikét



4. kép A házirend lefordítása



5. kép Általános házirend egy tűzfalhoz

naplózza. Valóban, én is inkább az eldobott csomagokat naplózom, az engedélyező szabályok naplózásától pedig eltekintek. A 2. képen látható mintaszabályok sora egy általános szabállyal zárul: ez az összes olyan csomagot eldobja, amelyik sem az előtte található szabályok valamelyikének, sem a felületekre külön megadott szabályoknak – például a helyi visszacsatolóra vonatkozó házirendnek – nem felel meg.
Ennek a szabálynak tulajdonképpen kizárólag a naplózásban van szerepe.
A Firewall Builder az alapértelmezett házirendet az összes IP Tables lánc esetében `DROP`-ra állítja, de ezeket az alapértelmezés szerint eldobott csomagokat a rendszer nem naplózza, hacsak a Netfiltert nem utasítjuk kifejezetten erre. Létezik már kísérleti jellegű folt a Netfilterhez, ami alapbeállítás szerint is lehetővé teszi az összes eldobott csomag naplózását, de szerintem érdemes megvárni a kód üzembiztossá válását, és csak azután fordítanám bele a rendszermagba. Ha ezt valamiért nem akarod megvárni, akkor ezt a szolgáltatást úgy érheted el, hogy a Firewall Builderben kiválasztod a tűzfalobjektumot, rákattintasz a hozzá tartozó ***Firewall Properties*** (Tűzfal tulajdonságai) fülre, majd bejelölöd a ***Log all dropped packets*** (Naplózzon minden eldobott csomagot) jelölőnégyzetet. Az említett foltról a ➲ http://www.netfilter.org/documentation/pomlist/pom-summary.html címen találhatsz további tájékoztatást.

A házirend lefordítása és telepítése

Miután a tűzfal házirendje elkészült, IP Tables-parancsfájllá kell alakítani. Ehhez először ellenőrizni kell, hogy az ablak bal oldali részén megjelenő fában a tűzfal objektuma és az általános házirend vagy a felületekhez rendelt saját házirendek valamelyike ki van-e választva. Ezt követően nyisd meg a ***Rules*** (Szabályok) menüt, és válaszd a ***Compile*** (Fordítás) pontját. Az eredmény a *3. képen* látható. A házirend sikeres lefordítása után a Firewall Builder egy fájlba írja az eredményt, aminek a neve annak a tűzfalnak a nevével egyezik meg, amelyhez a lefordított házirend tartozik, a kiterjesztése pedig *.fw*. A létrejött példaparancsfájl, a ***trillian.fw*** tartalma – a terjedelem miatt kissé módosítva, rövidítve – a *listában* látható. Az írásomban említett szabályok mindegyike megtalálható benne.
Az új parancsfájlt kézzel is átmásolhatod a Trillianre, majd változatlan formában futtathatod, vagy kézzel is lefordíthatod egy olyan indítási parancsfájlba, amely a Trillian Linux-terjesztésének megfelel, például egy szabványos Red Hat 7.3 indítási parancsfájlba. Könnyebb önműködő módon átmásolni és üzembe helyezni egy a Firewall Builderhez készült telepítő parancsfájl segítségével. Ilyenre példa az `fwb_install`, amely a ➲ http://sourceforge.net/project/showfiles.php?group_id=5314 címről érhető el. Utóbbi különösen elegáns és egyszerű módja a tűzfalszabályok biztonságos átmásolásának és üzembe helyezésének, az `fwb_install` ugyanis `scp`-n keresztül másolja a parancsfájlt a távoli gép */etc/firewall* könyvtárába, majd SSH-n keresztül indítja el. Ha az `fwb_install`-t már valahová letöltötted a Firewall Buildert futtató gépeden, akkor a Firewall Builderen belül az egyes tűzfalobjektumok ***Compile***/***Install*** (Fordítás/Telepítés) tulajdonságai között írhatod elő a használatát. Ne feledd az `fwb_install`-t a rendszered beállításainak megfelelően módosítani, illetve az általa használandó SSH-kulcsokról is gondoskodnod kell. Miután ezekkel végeztél, a szabályok lefordítás utáni telepítéséhez mindössze a ***Rules*** (Szabályok) menü ***Install*** (Telepítés) pontjára kell kattintanod.
Az `fwb_install` ügyes segítőtárs, ám szükséged lesz egy olyan indítási parancsfájlra is a célrendszeren, amelyik indításkor mindig futtatja a tűzfalparancsfájlt. Egyébként a rendszer védtelen lesz minden indítás vagy újraindítás és a között az időpont között, amikor a Firewall Builder `Install` parancsát lefuttatod. A saját rendszered */etc/init.d* könyvtárában található fájlok megfelelő mintául szolgálnak ehhez.

Házirend egy valódi tűzfalhoz

Írásom jelentős részét a bástyagépes példának szenteltem, ám szerencsére egy többlaki (több felülettel is rendelkező) tűzfal házirendjét is nagyon hasonló módon lehet összeállítani. A visszacsatoló szabályokkal kell kezdeni, a többi felületen meg kell akadályozni a hamisításokat, létre kell hozni egy általános házirendet, lefordítani és végül telepíteni. A legfontosabb különbség az, hogy egy

tűzfal – a kiszolgálók többségével ellentétben – több hálózati felülettel is rendelkezik. Mivel egy egyetlen felülettel rendelkező rendszer a visszacsatolótól eltekintve az összes csomagot ugyanazon a fizikai eszközön keresztül kapja, a hamisított csomagokat nem tudja megkülönböztetni a valódiaktól, és minden csomag forrás-IP-címét kénytelen elfogadni. Egy többlaki rendszer ezzel szemben könnyedén szét tudja válogatni azokat a csomagokat, amelyek ténylegesen a helyi hálózatról érkeznek, illetve azokat, amelyek valójában az Internet felől futottak be, de olyan hamis forrás IP-címmel, ami helyi vagy megbízható hálózatból jövőnek láttatja őket.

Példámban a belső hálózat a 192.168.111.0 (alhálózati maszk: 255.255.255.0) címet kapta. Ha van egy Slartibartfast nevű tűzfal, amely e hálózat és a külvilág között helyezkedik el, akkor hamisításellenes szabályokkal utasíthatjuk arra, hogy a belső hálózat felé vezetőtől eltérő felületeken azonnal dobjon el minden olyan csomagot, amelynek forrás IP-címe 192.168.111-gyel kezdődik. Az ilyen csomagok nyilvánvalóan hamisak.

A *4. képen* a Slartibartfast hamisításellenes szabálya látható.

Mielőtt ezt a szabályt létrehoztam volna, olyan hálózatobjektumokat készítettem, amelyek a 1918-as RFC-ben („Address Allocation for Private Internets”) foglalt, fenntartott IP-címtartományokat fedik le. Az RFC 1918 címtartományai kizárólag szervezeteken belüli használatra valók, az Interneten keresztül nem irányíthatók, így minden internetes tűzfalnak eleve feltételeznie kell, hogy az ilyen forráscímmel érkező csomagok hamisak – a 4. képen látható szabály pontosan erről gondoskodik. Mivel nálam az RFC 1918 szerinti C osztályú objektum a 192.168.0.0 tartományba esik (alhálózati maszk: 255.255.0.0), belsőhálózati címem pedig 192.168.111.0 (része az RFC 1918 által megadott címtérnek), a `Net_Internal` objektumot ebben a szabályban nem kellett szerepeltetnem. Egyébként, ha valaki nem ismerné a 1918-as RFC tartalmát, az RFC 1918 A osztályú objektumom a 10.0.0.0 címre (alhálózati maszk: 255.0.0.0), az RFC 1918 B osztályú pedig a 172.16.0.0 címre (alhálózati maszk: 255.240.0.0) hivatkozik.

Általános szabályok

Az *5. képen* a Slartibartfast általános házirendje látható, ennek részletes ismertetését mellőzném – mondandóm már így is túl hosszúra nyúlt. Mivel a Firewall Builder lényege éppen a tűzfalszabályok könnyen olvasható formában való megjelenítése, az 5. kép értelmezése talán nem okoz különösebb gondot.

Ha már a könnyű megértésről esik szó: talán még nem említettem, hogy a helyi visszacsatolóra vonatkozó, hamisításellenes és általános szabályok egyaránt könnyedén létrehozhatók a Firewall Builder szabályvarázslójával.

Remélem, bocsánatos bűn a részemről, hogy csak most említem a varázslót. A tűzfalszabályok azonban túl fontosak ahhoz, hogy elkészítésükkor holmi varázslóban vakon megbízzunk.

Linux Journal 2003. június, 110. szám

Mick Bauer (mick@visi.com) Biztonsági szakember, a Linux Journal biztonsági témákkal foglalkozó szerkesztője, biztonsági tanácsadó a Minnesota állambeli Minneapolisban található Upstream Solutions LLC Inc.-nél. Mick szabadidejében a gyermekeivel fogócskázik vagy zenél – néha mindkettőt egyszerre.

Tudományos folyamatok gyorsítása

Nézzük meg, hogyan tehetjük hatékonnyá Matlab projektjeinket egyes részeinek C-kóddá történő fordításával!

A NASA Moffett Field-i Ames Kutatóközpontjában, a Szilíciumvölgy szívében, egy olyan csapatban voltam kutató, amelyik Linuxot használt egy igen érdekes és előremutató fejlesztésben. Az Ames-i Neuro Engineering Laboratóriumban az agy–számítógép-csatolófelületen dolgoztam, mégpedig egy olyan rendszeren, amelynek a segítségével az EEG (elektroencephalogram–agyhullám) jeleket elektronikus rendszerek és roboteszközök vezérlésére lehet felhasználni. Az volt a feladatom, hogy az elsődleges labor kutatóitól átvegyem a prototípuskódokat, majd ezekből hatékony megvalósításokat fejlesszek ki, amelyeket valós idejű adatfeldolgozással, emberi alanyokon ki lehet próbálni. Gyakran csak az algoritmusok durva vázlatát vagy egy kódtöredéket kaptam meg, és ki kellett derítenem, hogy fel lehet-e őket használni az általunk gyűjtött agyhullámadatokon.

A Matlab és a GNU Octave nevű szabad program nagy segítséget jelentett nekem ebben a munkában: lehetővé tették, hogy az adatfeldolgozásra és adatmegjelenítésre vonatkozó hatékony módszereket dolgozzak ki, amit C-ben (ne adj Isten, Fortranban) csak igazán kínkeservesen tudtam volna megoldani. A megvalósítás egyszerűsége pedig komoly szempont olyankor, amikor nagy tömegű kísérleti kóddal dolgozunk, amiket aztán vagy véglegesen felhasználunk, vagy elvetünk. Amikor az eljárás megfelelőnek tűnt, és eljött az idő, hogy komolyan elgondolkozzunk valós idejű adatfeldolgozó rendszerünkbe történő illeszthetőségén, azonnal kiderült, hogy a Matlab könnyű programozhatóságának bizony megvan az ára. Ez az ár pedig a sebesség. Egyetlen másodpercnek megfelelő adatmennyiség feldolgozása például percekig vagy akár órákig is eltarthat. Nyilvánvaló, hogy ezt nemigen használhatjuk valós idejű rendszerekben. Minden kód, ami értékesnek bizonyul, meglévő kódbázisunkhoz illeszkedve előbb vagy utóbb C-, illetve C++-kódként végzi. E két ok miatt a Matlab elég sok kódját C nyelven újra kellett írnunk.

Aki már dolgozott Matlabbel, most biztosan rögtön arra gondol, hogy „de hiszen a Matlab magától is képes C-be menteni”, vagy „mi a baj a Matlab új JIT fordítójával?”. Az új JIT fordító helyenként valóban felgyorsíthatja a kódot (ha belenézünk a leírásba, látni fogjuk, hogy jó néhány olyan helyzet létezik, amikor meg sem próbál egyszerűsíteni), azonban soha nem versenyezhet egy jól megírt, lefordított C-kód hatékonyságával. A Matlab C-exportáló felületéről pedig csak annyit, hogy a Matlab által kiírt kód pontosan ugyanolyan lassú, mint a Matlab belső környezetében futó értelmező (interpreter) kód, és csatolófelület-munkálatok nélkül igen nehéz más projektekbe illeszteni. Továbbá egyik lehetőség sem segíti a GNU Octave felhasználóit vagy azokat, akik nem tudják tartani az ütemet a költséges Matlab-frissítésekkel. Általában véve a legjobb megoldás az, ha az eredetileg Matlab alatt létrehozott algoritmusainkat kézzel alakítjuk át gyors, termelési szintű kóddá.

Ebben a cikkben először néhány tippet láthatunk, hogyan írhatunk némileg hatékonyabb Matlab-kódot. Majd bemutatjuk, hogy miképpen lehet a MEX-függvények segítségével C-kódot illeszteni a Matlab programba, hogy felgyorsítsuk a végrehajtást, miközben továbbra is Matlab-környezetben maradunk. Innen már csak egy viszonylag apró lépés a teljes projekt átvitele C vagy C++ alá. Az itt található adatok különböző helyeken a hálózaton is megtalálhatóak; ezt a cikket inkább rövid HOGYAN-nak, még inkább személyes jegyzetnek szántuk – olyannak, ami bemutatja, hogyan hozhatunk át egy kísérleti Matlab-kódot a valódi világba.

Cikkünkben példaként a fej felületén mért feszültség gyors változásainak szétválasztására tervezett kódot fogjuk használni. A kód a többelemű eseményfüggő potenciálbecslés (multicomponent event-related potential estimation, röviden mcERP) nevű algoritmuson alapszik. Akkor kezdtem el először Matlab-programok C-kóddá alakításával foglalkozni, mialatt ezen az algoritmuson dolgoztam. Amikor ugyanis valamilyen beállítási értékkel és bemenő adatokkal kipróbáltam az algoritmust, általában egész éjszakára ott kellett volna hagynom. Semmilyen Matlabba épített optimalizálási módszer sem volt képes jelentősen csökkenteni a végrehajtási időt.

A teljes C-átalakítást követően általában tíz másodperces nagyságrendű időmennyiségre volt szükség egy nagyobb adatmennyiség feldolgozásához. Ez a különösen nagy időmegtakarítás az algoritmus igen mélyen egymásba ágyazódó jelle-

1. lista Beágyazott ciklus a mcERP algoritmusban

```
for i = 1:M
   for j = N:-1:1
      norm = 0;
      for r = 1:R
         for t = 1:T
            X = x(i,r,t);
            for n = 1:N
      if (n ~= j)
          X = X - coupling(i,n) *
                      alpha(n,r) *
                      s (n,pad+t-tau(n,r));
      end
            end
         Y = alpha(j,r)*s(j,pad+t-
             ➥tau(j,r));
         c(i,j) = c(i,j) + X*Y;
         norm = norm + Y^2;
       end
    end
    c(i,j) = c(i,j)/norm;
  end
end
```

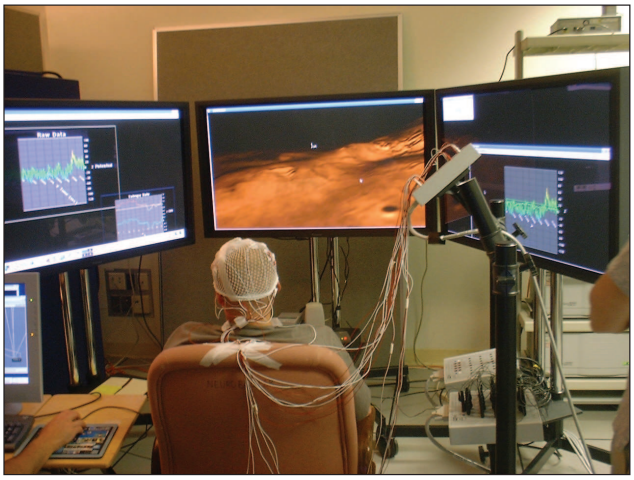
gével magyarázható (lásd az *1. listát*). Nem állíthatjuk, hogy minden algoritmus ennyire felgyorsul. Ráadásul még ez a teljesítmény sem elég jó a valós idejű műveletekhez, de már elég gyors ahhoz, hogy adatcsökkentő módszereket, kódpár-huzamosítási megoldásokat és egyéb trükköket kezdhessünk el keresni, hogy a kívánt sebességhez közeli értéket érjünk el.

Kódoegyszerűsítés Matlabben

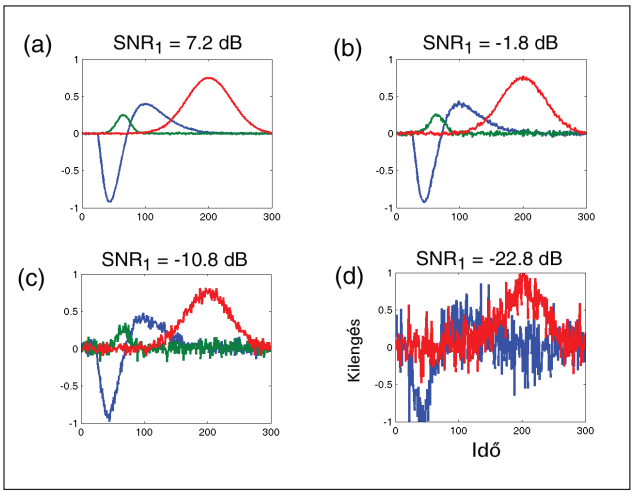
A Matlab teljesítménye a ciklusok futtatáskor a leggyengébb. Akár azt is mondhatnánk, hogy a Matlab kifejezetten utálja a ciklusokat; sokkal hatékonyabban tud végrehajtani néhány ciklusszerű műveletet, ha a kódot előtte vektorizáljuk, és a függvényeket mátrixba rendezett adathalmazon hajtjuk végre, mintha végig kellene lépkednie az adatokon. Sajnos ez csak bizonyos műveletek esetében működik. Amikor nagyobb dimenziószámú mátrixokkal dolgozunk, gyakran kapunk nehezen olvasható és érthető kódot. A ciklusképzés viszont éppen az a terület, amiben a C igencsak jeleskedik – mutatóműveletekkel végiglépkedni egy mátrixon (amennyiben nagy adathalmazokkal dolgozunk) hihetetlenül hatékony és gyakran a legérthetőbb módszer. A Matlab-kódok C-optimalizálásakor a legtöbb energiát a beágyazott ciklusszerkezetek egyszerűsítésébe kell fektetnünk.
Egyéb módszerek, amelyek segítségével megnövelhetjük a Matlab hatékonyságát:

1. Minden tömböt, még a kisebb méretű tömböket is, értékadás előtt a `zeros()` függvény segítségével előre foglaljunk le, és ne hagyjuk, hogy a Matlab az értékadás során egy már létező tömbhöz adatokat rendeljen.
2. Ahogy azt a Matlab leírásában is olvashatjuk, parancsfájlok (scripts) helyett minden kódunkat függvényekben érdemes tárolni. Ezáltal két–háromszoros sebességnövekedést érhetünk el.
3. Szervezzük az adatainkat úgy, hogy a mátrixműveletek oszlopcentrikus módon hajtódjanak végre. A Matlab Fortran-stílusban tárolja a tömböket, azaz a mátrixok oszlopai folytonos területet foglalnak el a memóriában. Ez C alatt pont fordítva működik: itt a memóriában a mátrixok sorai helyezkednek el folytonosan. Amennyiben valamilyen adathalmazon szeretnénk egy függvényt végrehajtani, az adatokat oszlopban, és ne egy mátrix sorában tároljuk. Lehet, hogy ez teljesen értelmetlen és furcsa számunkra, de úgy tűnik, akad valami haszna.
4. Próbáljuk meg elkerülni a belső típusátalakításokat, amelyek időről időre megtörténnek. Ez is egy olyan példa, amire nincsen igazi bizonyítékom, de a Matlab általában nem kényszerít bennünket arra, hogy megadjuk a változók adattípusát, és néha igen könnyű olyan ciklust létrehozni, amelybe implicit típusátalakítás kerül. Sokkal jobb megoldás, ha változóinkat az ismételt végrehajtás előtt általános adattípussá alakítjuk. Tulajdonképpen ez ugyanaz, mint amit C vagy C++ nyelven tennénk, csak éppen nehezebb a helyes utat látni, mivel Matlab alatt a változók közvetlenül szinte soha nem kapnak típust.

Vessünk egy pillantást az mcERP algoritmus kódjának egy szeletkéjére (1. lista). Itt bemutatjuk a kódba ágyazott számos ciklus egyikét. Az mcERP algoritmus Bayes-féle összetett bejárásos (iteration) hullámformabecslésen alapul. Számos, az itt látható részlethez hasonló ciklus található a kódban, amiket többször le kell futtatni, hogy megfelelhessenek az adatban fellelhető hullámformáknak.



1. kép
Ezen a fotón kísérleti összeállításunkat láthatjuk, ahol a valós visszajelzésű agy–számítógép-csatolófelület kifejlesztését célzó kísérleteket végezzük. A három nagy megjelenítőn keresztül tökéletesen irányítani tudjuk, hogy az alany mit láthasson látóterületének legnagyobb részén. Az összes számfeldolgozó és megjelenítő program házon belül készült, és Linux alatt fut



1. ábra
Az ábrán az mcERP algoritmus példakimenetét mutatjuk be, ahol fejtető-elektródákból kiolvasott valós idejű potenciálból levezetett alap-hullámalakok láthatók egy szimulált kísérleti folyamatban. Minden egyes hullámforma számos, egyre pontosabbá váló Bayes-féle hullámforma-közelítési ciklus eredménye, ahol minden egyes ciklus egy ütem alatt rengeteg számítást végez. Ezeket az eredményeket Matlab alatt hosszú órák alatt kapjuk meg, ugyanakkor a számítás csak másodperceket, esetleg perceket vesz igénybe, ha az algoritmus egyes részeit C nyelven írjuk újra

Láthatjuk, hogy az ilyen típusú szerkezetek nem futnak túl gyorsan az értelmezőben, hiszen az a ciklusok futtatásakor nem teljesít valami jól. Ugyanakkor a belső `if` utasítás miatt a kódot belső függvényhívás hozzáadása nélkül nem tudjuk vektorizálni – így ez a módszer nem sokat segít. Ezért aztán C-átalakításunkhoz ez a kód lesz az elsődleges tesztalany. Természetesen ettől még nem baj, ha az algoritmusképzés során a Matlabot is felhasználjuk, hiszen igen könnyen készíthetünk vele az *1. képhez* hasonló látványos képeket. A MEX-

2. lista MEX-függvény

```
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include "mex.h"
#include "pops.h"
void latcalc(double *x, double *erp,
             ➥double *alpha,
             double *tau, double *coupling,
             double *range,
             ➥double *inresultlat,
             ➥double *inresulterp,
             int M, int R,
             int T, int NsoFar);

void mexFunction(int nlhs, mxArray * plhs[],
                 int nrhs, const mxArray *
                 ➥prhs[])
{

    int M, R, T, NsoFar;
    const int *dim_array;

    if ((nrhs != 6) || (nlhs != 2))
        mexErrMsgTxt("wrong # of args to
                     ➥c_mLAT");

    dim_array = mxGetDimensions(prhs[0]);

    M = dim_array[0];
    R = dim_array[1];
    T = dim_array[2];

    NsoFar = mxGetM(prhs[1]);

    plhs[0] =
        mxCreateDoubleMatrix(NsoFar, R,
        ➥mxREAL);
    plhs[1] =
        mxCreateDoubleMatrix(NsoFar, T,
        ➥mxREAL);


/*A tényleges munkát végző függvény
meghívása*/
    latcalc(mxGetPr(prhs[0]),
            ➥mxGetPr(prhs[1]),
             mxGetPr(prhs[2]),
             mxGetPr(prhs[3]),
             mxGetPr(prhs[4]),
             mxGetPr(prhs[5]),
             mxGetPr(plhs[0]),
             mxGetPr(plhs[1]),
             M, R, T, NsoFar);


}
```

függvénynek nevezett megoldást fogjuk használni. Ezzel a módszerrel a belső mag gyors futásra lesz képes, ugyanakkor megtartjuk a teljes algoritmus finomítását és vizsgálatát végző Matlab-részekhez kapcsolódó csatolófelületet.

C-környezet összeállítása Matlab alatt

A MathWorks honlapján (lásd a ***Kapcsolódó címeket***) igen jelentős mennyiségű leírást találunk, amit a cikk elolvasása után nem árt, ha átnézegetünk, amennyiben komolyan MEX-függvényekkel szeretnénk foglalkozni. Ez a cikk az egyszerűsítésre összpontosít, illetve néhány lényeges pontot emel ki, amelyek segítenek életet lehelni a dolgokba.
Ha Linux alatt szeretnénk MEX-függvényeket fejleszteni, térjünk vissza minden jó forrásához, a parancssorhoz, és gépeljük be a `mex start` utasítást. Ha ez nem működne, a Matlab telepítési könyvtárában keressük meg a MEX-parancsfájlt. Elképzelhető, hogy elérési utunkban rendszergazdánk csak a futtatható Matlab-állományra készített hivatkozást.
A MEX futtatása során kiválaszthatjuk a fordítónkat. Hogy kihasználhassuk a későbbiekben ismertetett fordítófüggő egyszerűsítési lehetőségeket, jobb, ha a Matlab beépített LCC rendszere helyett inkább a GCC-t használjuk. A MEX ezután létrehozza a ***~/.matlab/R12/mexopts.sh*** állományt, amit akkor használ fel, amikor a MEX-eszközzel külső kódot fordítunk a Matlabhez. Hasznos és tanulságos belenézni a ***mexopts.sh*** állományba felület/fordító párosunk megfelelő szakasza alatt. x86 Linux/GCC használata esetén nézzük át a fő elágazás `glnx86` szakaszát. Minden változtatás, amit ezen a szakaszon kívül végzünk el, semmilyen hatással sem lesz a kódfordítás menetére. Helyezzünk ide azokat a kapcsolókat, amelyekkel C-függvényeinket fordítani szeretnénk. Amennyiben szeretnénk hatékonnyá tenni, a következőket használhatjuk:

```
COPTIMFLAGS='-O3 -funroll-loops -finline-
functions'
```

(ami elég kíméletlen – úgyhogy legyünk óvatosak), illetve bármilyen más tetszés szerinti kapcsolót is használhatunk. Ha a fordításánál ezeket a kapcsolókat szeretnénk használni, a MEX-et a `-O` kapcsolóval kell lefuttatnunk. Akárcsak a `make`-fájlok esetében, szúrjuk be ide azokat a fejléckönyvtárakat, amelyeket a `CFLAGS` végéhez szeretnénk hozzácsapni:

```
-I/a/fejléc/mappa/útvonala
```

A befűzendő programkönyvtárakat így jelöljük:

```
-l[lib-név]
```

és:

```
-L/a/programkönyvtár/mappa/útvonala
```

Ezek az `LDFLAGS` végére kerülnek.
Ha ezzel megvagyunk, készítsünk egy könyvtárat, ahol majd a MEX-szel fordítandó C-fájljainkat tartjuk. Nem javaslom, hogy C alapú és Matlab alapú kódjainkat azonos könyvtárban tároljuk. Ezt a könyvtárat, vagy az általunk készített harmadik eredmény- (build) könyvtárat adjuk a Matlab-környezet elérési útjához. Most már készen állunk a kód létrehozására.

Hogyan írjunk C-kódot Matlab alatt?

Mindenekelőtt gondoljuk át, milyen céljaink vannak az egyszerűsítéssel, illetve hogy programunk mely részei nyerhetnek a legtöbbet azáltal, ha C nyelven újraírjuk őket. Mivel a C elsősorban a ciklusok értelmezése terén lényegesen jobb a Matlabnál, logikusnak tűnik, ha végignézzük a kódunkat, és kikeressük, hol alkalmaztunk jelentős mennyiségű adattal dolgozó ciklusokat. Nemigen éri meg újrakódolni valamit, ami mindössze háromszor hajtódik végre, de ha például egy 500×500×500 egységes térben lépkedünk végig minden egyes voxelen, a C-kód rengeteg megtakarítást jelenthet. Külön figyelmet érdemelnek a beágyazott ciklusokban található egyszerű műveletek, ilyeneket az 1. lista kódrészletében is láthattunk. Ami összetett műveleteket végez egy beágyazott ciklusban – azaz amit láthatóan csak nagyon nehezen tudnánk magunktól elkészíteni, vagy nem találunk hozzá egy harmadik fél által készített programkönyvtárat –, az valószínűleg nem lesz jó kiindulási pont az egyszerűsítéshez. A Matlab-függvényeket C-kódból is meg tudjuk hívni, de ezzel nyilvánvaló módon nem sokat javítottunk a végrehajtási időn.

A C MEX-állományok készítését általában azzal kezdjük, hogy az algoritmusunk valamelyik kódblokkját függvényesítjük, esetleg egy már korábban elkészült függvényt jelölnünk ki egyszerűsítésre. Ha ez megvan, eljött az ideje, hogy elkészítsük a függvény C-változatát. A módszer a következő: hozzuk létre az általános Matlab-csatolófelület-függvényt, majd a tényleges munkát végző lényegi függvényt. A *2. listában* egy ilyen MEX-függvényre láthatunk példát. Az itt meghívott lényegi függvény az 1. listában látható Matlab-fájlhoz tartozik, és a Linux Journal FTP-lapján az ➲ ftp.ssc.com/pub/lj/issue110/6722.tgz címen.

A `mexFunction()` a MEX-fájlprogramozás világában olyasmi, mint C-ben a `main()`. Amikor Matlab alól meghívjuk a függvényünket, ez a függvény fog elindulni. Függvényünk tényleges nevét a lefordított *.c*-állományok neve határozza meg, pontosabban általában a MEX-nek fordításkor elsőként átadott *.c*-állomány neve. Linux x86 felületen a MEX-fájlok kiterjesztése ***.mexglx***. Ha a Matlabet Linux x86-felületen futtatjuk, a Matlab ugyanúgy és ugyanazon az elérési úton keresi a ***.mexglx***-állományokat, mintha a hagyományos *.m*-fájlokat keresné, így az ***.mexglx***- és *.m*-állományok felcserélhetők. Ügyes módszer a Matlab és az egyszerűsített kód közötti váltásra, ha megváltoztatjuk a Matlab keresési útvonalát. Én például a ***c_mLAT.c*** állományt ***c_mLAT.mexglx*** fájllá fordítottam, majd a lefordított kódot a Matlab-környezetből egyszerűen a `c_mLAT()` függvénnyel használhatom. Remek kis rendszer.

A dolgok akkor kezdenek egy kicsit bonyolultabbá válni, amikor adatokat szeretnénk át- és visszaadni a Matlab és a C között. A `mexFunction` tulajdonságai (argumentum) közt két duplamutatót vehetünk észre. A `*plhs[]` a függvény visszatérési értékeire mutat (a Matlab-függvényeknek több visszatérési értékük is lehet), a `*prhs[]` pedig a bemenő értékeket tartalmazza. A bemenő adatok és a visszatérési értékek számát az `nlhs` és az `nrhs` értékekben szintén átadhatjuk. A `mexFuntion()` függvényen belül az `mxCreateDoubleMatrix()` függvénnyel kell lefoglalnunk a visszaadott mátrixok memóriahelyét, amennyiben helyesen akarjuk őket a Matlab-környezetnek visszaadni. Az mx-függvények a Matlab-környezetben készítenek memóriaterületet és a Matlab memóriakezelője kezeli őket, így az mx-függvények által foglalt területek felszabadításával nem kell foglalkoznunk. A mex szócskával kezdődő függvények a Matlab-környezetben futnak le; a `mexCallMATLAB()` függvénnyel például tetszőleges Matlab-függvényt hívhatunk meg a programunkból. A `mexFunction()` függvényünk belsejéből a kimenet lefoglalása után meghívhatjuk a lényegi munkát végző függvényünket, formára szabhatjuk a bemenetet, és például ellenőrizhetjük az értékeket.

A Matlab-leírás sajnos nem foglalkozik egy, a C-programozók számára igen kiábrándító nehézséggel, nevezetesen azzal, hogy a Matlab az adatait oszlopcentrikus formában tárolja. Ez különösen bosszantó lehet, hiszen amikor végiglépkedünk a többdimenziós bemeneti mátrixokon, nyilván a könnyen érthető mutatóaritmetikát szeretnénk használni. Viszont meglehetősen kiábrándító lehet, és hibalehetőségeket rejthet magában, ha nekünk kell kitalálnunk, hogy ciklusonként milyen messzire kell ugranunk és így tovább. Amennyire én látom, három megoldás létezik:

1. Számoljuk ki magunk, hogy ciklusonként mekkora lépéseket kell megtennünk, és tartsuk észben. Ez megoldhatónak tűnik, és talán a legjobb megoldás, de három- vagy több dimenziós mátrixok esetében kétségtelenül komoly fejfájást okoz.
2. Formázzuk újra a kódot, mielőtt belépnénk a MEX-függvénybe, hogy a C nyelvnek megfelelően legyen szervezve. Igaz, ez Matlab alatt igen költséges lehet, és valószínűleg nem árt, ha van egy helyettesítő példányunk az eredetiből.
3. Tegyük azt, amit én is tettem, és készítsünk makrókat vagy makrószerű elemeket, amelyekkel a Matlab-tömböket elérhetjük. Ez lassabb ugyan, mint végiglépkedni a tömbökön, és talán nem is tűnik túl elegáns megoldásnak. Tapasztalataim szerint azonban általában igen gyors és a kódja jól olvasható. Én például egy ***pops.h*** nevű állományt készítettem, ami a következő függvényt tartalmazza:

```
extern inline double
num3d (double *start,
       int rows, int cols,
       int x, int y, int z)
{
     return(*(start
              + rows * cols * z
              + rows * y
              + x));
}
```

KAPCSOLÓDÓ CÍMEK

The MathWorks, a Matlab készítője
➲ http://www.mathworks.com
A Matlab dokumentációs forrása
➲ http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml Válasszuk az External Interfaces/API pontot, majd innen a MEX-leírást.
A Matlabhoz hasonló GNU Octave szabad program honlapja ➲ http://www.octave.org
A NASA Ames-i Számítási Tudományos Részleg, a NASA Neuromérnöki Laboratórium bázisa
➲ http://ic.arc.nasa.gov

A fenti részlet 3D Matlab-tömbformátumban adja vissza az értékeket, amennyiben megadtuk a tömb címét, a sorok és az oszlopok számát, valamint a lekért adat xyz helyzetét. Kicsit talán esetlen, de azért használható. A kódoptimalizálás során ez a kódrészlet a programban ugyanúgy beszúródik, mint az előfeldolgozó makrók. Makrókat is alkalmazhatunk, de én ezt a módszert sokkal könnyebben használhatónak és nyomon követhetőnek találtam. Szerencsére az, hogy a ciklusokat C-ben készítjük el, messze többet nyom a latban, mint tömbelérési módszerünk viszonylag kis vesztesége.

Egyéb tekintetben a MEX-fájlok készítése nem túl bonyolult feladat. Amikor eljön a fordítás ideje, a MEX-programot azokkal a C-fájlokkal futtassuk, amelyeket le szeretnénk fordítani. A MathWorks weblapján megtaláljuk a MEX-fordítási lehetőségek listáját. Miután az ***X.c Y.c Z.c*** fájlokat lefordítottuk, egy ***X.mexglx*** állományt kapunk, amit aztán (amennyiben az elérési utunkban található) a Matlab parancssorából `X()` néven használhatunk.

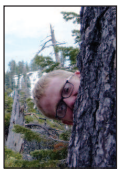
Mostantól kezdve kódunk egyre nagyobb és nagyobb részeit írhatjuk át C-be. Amikor aztán eljön a teljes C-megvalósítás ideje, a külső Matlab-kód átalakítására gyakran megéri a Matlab C-exportálási képességét használni, hiszen a lényeges részeket korábban már amúgy is egyszerűsítettük. Amennyiben a dolgok még mindig nem elég gyorsak, érdemes lehet a külső függvényt újraírni, hogy a memóriát C-barát módon kezelje. Ezáltal felgyorsíthatjuk a belső C-kód ciklusait, hiszen a tömböket egyszerűsített mutatólépegetéssel érhetjük el.

Általában véve jelentősen felgyorsíthatjuk a dolgokat, ha Matlab nyelven fejlesztünk és Matlab-prototípusokat használunk. Előfordulhat azonban, hogy éjszakákat kell várnunk, mire a Matlab kihozza az eredményt, majd kiderül, hogy elgépeltünk egy apró bemeneti értéket. Nos, az ilyesmit elkerülendő a kézzel egyszerűsített kódrészletek igencsak hasznosak lehetnek, ha algoritmusunkat célszerűen szeretnénk használni.

Irodalomjegyzék

Shah, A. S.-Knuth, K. H.-Truccolo, W. A.-Ding, M.-Bressler, S. L.-Schroeder, C. E.: A Bayesian approach to estimating coupling between neural components: evaluation of the multiple component event-related potential (mcERP) algorithm. Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 22nd International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, C. Williams, American Institute of Physics, Melville, New York, 2003

Linux Journal 2003. június, 110. szám



Sam Clanton

Jelenleg MD/PhD-tanuló a Pittsburgh/Carnegie Mellon Egyetemen, kapcsolatot tart fenn a NASA-val és a kaliforniai Mountain View-i QSS Corporation Ames Kutatóközponttal. Idejét biológiai jelfeldolgozási feladatok megoldásával, számítógépes látási és orvosi robotikai vizsgálatokkal tölti, és nagyon érdekli a természetben létrejött rendszerekhez hasonló informatikai rendszerek összeállítása. Sam idejének javát leveleinek átnézésével tölti, és túl sok kávét iszik.

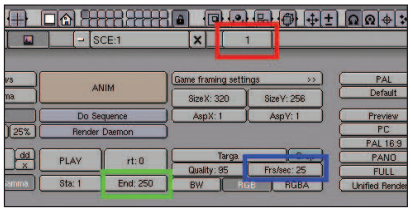
Animációkészítés a Blenderrel

Ebben a részben a Blender kiterjedt animációs lehetőségeivel fogunk foglalkozni.

Három módszert szeretnék bemutatni, és most is az egyszerűbbtől haladunk a bonyolultabb megoldások felé. Itt már belekóstolunk a karakteranimáció alapelemeibe, hogy a későbbiekben ne okozzon gondot az inverz kinematika lényegének és beállítási lehetőségeinek a megértése. Mielőtt azonban rátérnénk e hónap valódi témájára, az előző részhez kapcsolódóan egy ötlettel szeretném támogatni a felületi mintázatok leendő mestereit.

Amennyiben már van némi elképzelésünk a tárgy mintázatáról és arról is, hogyan fogjuk a mintázatot a tárgyra feszíteni, azt kézzel megrajzolhatjuk, szépen kifesthetjük, majd egy képolvasóval (scanner) újradigitalizálva nagymértékben megkönnyíthetjük a munkánkat. Természetesen a dolog fordítva is működhet, vagyis miután az üres képen elhelyeztük a háromszögeket, a képet kinyomtatjuk, kifestjük, majd ismét beolvassuk, és ezt a színes képet használjuk mintázatként. Ezek egyszerű ötletek, de úgy tűnt, hogy a mintázatok elkészítése könnyebben megy kézi eszközökkel, mintha egérrel kellene pontosan elhelyezni a megfelelő árnyalatokat, és a különböző időben létrehozott színeket újra és újra egymáshoz igazítani.

Ennyi kiegészítés után térjünk vissza eredeti témánkhoz, az animáció elkészítéséhez. A legegyszerűbb megoldás a kulcskocka alapú animáció. Hozzunk létre egy síklapot és felette egy gömböt a már megtanult módon, azaz a Space billentyű hatására megjelenő menü segítségével, majd az animáció képkockáinak számát állítsuk 42-re. Az animáció időtartamának beállításához az F10 billentyűvel kapcsoljunk a megjelenítést szabályozó nézetre, ezután az *1. képen* zöld színnel jelölt beállítómezőbe írjuk be a képkockák számát. Az animáció lejátszási idejét olyan módon számíthatjuk ki, hogy a képkockák számát elosztjuk a másodpercenként megjelenítendő képek számával, ami alapesetben huszonöt. Természetesen ezt is beállíthatjuk, ha az első képen kék színnel keretezett mezőbe



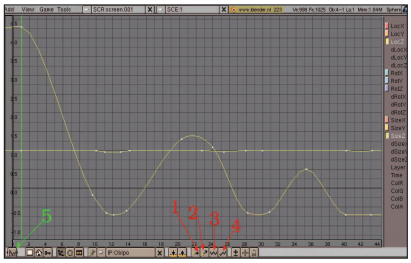
1. kép Az animáció beállításai

beírjuk a kívánt értéket (1. kép). Miután ezeket a beállításokat elvégeztük, ideje meghatároznunk a kulcskockákat. Egy-egy kulcsot minden esetben az ***I*** billentyűvel hozhatunk létre, a billentyű hatására megjelenő menüből választva ki a megfelelő kulcstípust. A következő választási lehetőségeink vannak egy-egy tárgyra vonatkozóan: a ***Loc*** alkalmazásával csak a tárgy helyét használjuk kulcsként, a ***Rot*** alkalmazásával a tárgy elforgatásának értékét, míg a ***Size*** a méretet jelenti. A következő két menüpont az előzők kombinációjának tekinthető, míg a ***Layer*** hatására azt tárolhatjuk kulcsként, hogy a tárgy melyik rétegen található. Ezt az utóbbi lehetőséget használhatjuk arra, hogy az animáció során egy tárgyat hirtelen eltüntessünk vagy megjelenítsünk.

Néhány szót szólnék a kulcskockák jelentőségéről. A Blender – sok más animációs programhoz hasonlóan – csak a kulcskockák által meghatározott tulajdonságokat tárolja, majd a közbeeső értékeket közelítéssel (interpolációval) számítja ki. A későbbiekben látni fogjuk, hogy ezeket a közbenső számításokat az úgynevezett ***IPO Editor***-ban pontosan szabályozhatjuk.

Tehát a következő lépés az lesz, hogy létrehozzuk a kulcsokat, és meghatározzuk a köztes értékek kiszámításának a módját. Célunk egy pattogó labda modellezése, most azonban csak a mozgásra fordítunk figyelmet, eltekintve az anyagtulajdonságok és mintázatok meghatározásától. Jelöljük ki a gömböt, majd az ***I*** billentyűvel az első képkockán hozzuk létre az első kulcsot, és a menüből válasszuk ki a ***LocRotSize*** pontot. Ezután a pillanatnyi képkockát állítsuk például 10-re, az első képen piros színnel jelölt mezőben. Helyezzük el a gömböt a síklapon, majd az előzővel megegyező típussal ismét hozzunk létre egy kulcsot. A pillanatnyi képkockát állítsuk például 12-re, és a gömb függőleges irányú átméretezése után készítsünk újabb kulcskockát. A 14-es képkockán a gömb ismét legyen eredeti méretű, és itt is készítsünk egy kulcsot az ***I*** billentyűvel. Ezután a gömböt mozgassuk ismét a síklap fölé, de most már ne emeljük ugyanolyan magasra, hiszen a pattogás során nem képes ugyanolyan magasra emelkedni. Ez legyen tehát a huszadik képkocka, majd hozzunk létre egy kulcsot, és a huszonhatodik képkockára lépve gömbünk megint a síklappal érintkezzen. Itt ismét hozzunk létre egy kulcskockát. Ismételjük meg a függőleges méretcsökkenést és növekedést oly módon, hogy a legkisebb méret a huszonnyolcadik képkockánál legyen, az eredeti méret pedig a harmincadiknál. Ne felejtsük el a kulcsokat is meghatározni! A harmadik emelkedés ismét kisebb magasságú legyen, mint az előző volt, ismételjük meg a méretcsökkenést és -növekedést a kulcskockák létrehozásával együtt. Az utolsó méretnövekedés vége a negyvenkettedik képkocka legyen. Ha mindezzel elkészültünk, a SHIFT-F6 billentyűkkel váltsuk át az egyik nézetet az ***IPO Editor***-ra. Ezt a nézetet úgy is megjeleníthetjük, hogy a nézet bal alsó sarkában látható ikonsorból kiválasztjuk azt az ikont, amit a *2. képen* az ötös számmal jelöltem meg. Ha mindent jól csináltunk, akkor ebben a nézetben a 2. képen láthatóhoz hasonló görbéket találunk. Ez a szerkesztő alkalmas arra, hogy elvégezzük a kulcskockák közötti értékek kiszámításának finomabb beállítását és meghatározzuk a kiszámítás módját. A nézet jobb oldalán látható a különböző színű görbék jelentése. A középső egérgomb lenyomva tartása mellett az egérrel mozgathatjuk az éppen megjelenített területet, míg a CTRL billentyű lenyomva tartása mellett a középső egérgombbal a nézet nagyítását változtathatjuk függőleges és vízszintes irányban (2. kép).

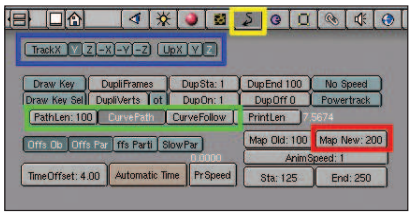
Az egyes görbéket itt is a jobb egér-



2. kép Az animáció finomhangolása

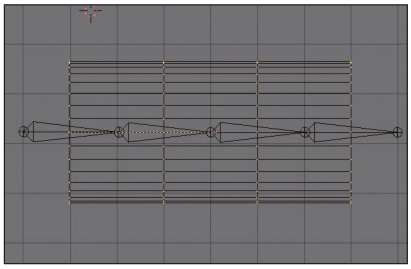
gombbal választhatjuk ki, majd a TAB billentyűvel pontszerkesztő módba válthatunk át, ahol a görbe alakját pontosan beállíthatjuk. Pontszerkesztő módban az egész pontot mozgathatjuk és elforgathatjuk, de kijelölhetjük az egyik vagy a másik végét is – ezzel a görbe adott szakaszának a meredekségét állíthatjuk be. Itt tehát állítsuk be pontosan a csillapodó mozgást, és kapcsoljunk vissza szerkesztőnézetbe. Ekkor már látható valamiféle pattogó labda, amit az ALT-A billentyűkkel mozgásra késztethetünk. Ez a billentyű szolgál az animáció előnézetének megjelenítésére.
Térjünk vissza az előző nézetbe, hiszen ahogyan a 2. képen is látható, akad még néhány érdekes beállítási lehetőség a mozgás leírására. Meghatározhatjuk, hogy az egyes kulcskockák között a Blender milyen módon számítsa ki a köztes értékeket. A 2. képen piros 1-es számmal jelöltem az alapértelmezett beállítást, ami szerint a Blender a kulcskockák között lineáris interpolációval számítja ki a közbeeső értékeket. A 2-es számmal jelölt ikon használatával a Blender az adott kulcs által meghatározott értéket kiterjeszti, vagyis a görbe meredeksége nem tér majd el a kulcskocka által meghatározott meredekségtől. A 3-as számmal jelölt ikonnal a Blender ismételni fogja a meglévő kulcskockák közötti görbeszakaszt, míg a négyes számmal jelölt ikon az előző kettő kombinációja, vagyis az ismétléseket a Blender a megadott szakasz kezdő- és végpontjához illeszti.
Természetesen nemcsak tárgyakhoz rendelhetünk kulcsokat, hanem az anyagtulajdonságokhoz is, így elérhetjük, hogy az animáció során egy tárgy színe vagy más tulajdonsága is megváltozzon. Ezeknek a lehetőségeknek könnyedén a nyomára bukkanhatunk, ha az anyagtulajdonságok nézetében a ***I*** billentyűt alkalmazzuk.
Térjünk át a következő megoldásra, amivel egy tárgyat rávehetünk arra, hogy egy előre meghatározott útvonalon haladjon végig.

Először készítsünk el egy útvonalat a ***Főmenü-Add-Curve-Path*** menüpontok kiválasztásával, majd ezt pontszerkesztő módban formáljuk a kívánt alakra. Ezután készítsük el, vagy töltsük be a tárgyat, ami majd ezen a görbén fog végighaladni. Ezek után már nincs más dolgunk, mint a CTRL-P billentyűk révén tudatni a Blenderrel, hogy ez az útvonal lesz a tárgy szülőobjektuma, majd az esetleges kezdőpozíció-eltérést a ALT-O billentyűk használatával el kell tüntetnünk. Így már a tárgy kiinduló helyzete megegyezik az útvonal kezdőpontjával. A szülő–gyermek-viszony meghatározásakor vigyázzunk, hogy a többszörös kijelöléskor a második legyen a szülő, ebben az esetben a görbe, különben a tárgy sehová sem megy az animáció lejátszásakor. E viszony meghatározására szolgál a CTRL-P billentyűkombináció, míg megszüntetése az ALT-P billentyűkkel lehetséges. A görbe követésének módját szintén szabályozhatjuk a Blenderben. Ehhez a F7 billentyűvel kapcsoljunk az animációs nézetbe, és tekintsük meg az alább látható *3. képet*.

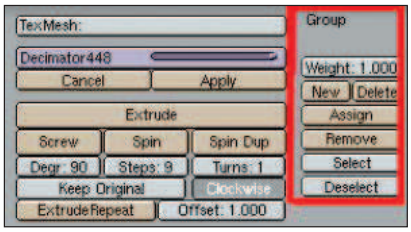


3. kép Az útvonal követése

Természetesen ezt a nézetet az ikonsorból is kiválaszthatjuk, ezt az ikont a képen sárga kerettel jelöltem. Fontos még meghatároznunk, hogy az útvonal követése során a tárgynak melyik tengelye nézzen előre és melyik felfelé. Erre szolgálnak a képen kék kerettel jelölt gombok. A gombok bal oldali része az előre néző tengelyt határozza meg, míg tőlük jobbra találhatóak a felfelé mutató tengelyt meghatározó kapcsolók. Természeten más módon is létrehozhatunk útvonalat a programban. Miután egy tetszőleges görbét létrehoztunk, azt az animációs nézetben útvonallá alakíthatjuk. A 3. képen zöld színnel bekeretezett területen található a `CurvePath` kapcsoló, amivel a kijelölt görbét útvonallá alakíthatjuk át. E kapcsoló mellett kapott helyet a `CurveFollow` beállító mező, amit ha bekapcsolunk, akkor az útvonalat követő tárgy tengelye idomul majd az út görbületéhez. Ezért tartottam fontosnak az előbbiekben megemlíteni a kék keretben lévő kapcsolókat a tengelyek beállításához. Nem hagyhatjuk figyelmen kívül azt az igényt sem, ha valaki nem pontosan száz képkockát szán arra, hogy egy tárgy végighaladjon a kijelölt úton. Szerencsére az animációs nézetben ezt is beállíthatjuk, miután a 3. képen piros kerettel jelölt beállítómezőben meghatározzuk a kívánt képkockák számát.
Felmerülhet a kérdés, hogy hogyan készíthetünk olyan mozgást, ami nem a teljes animáció kezdetén kezdődik, hanem a folyamat közben indul el. Ezt úgy oldottam meg, hogy amíg a tárgynak nem volt szabad mozognia, addig egy másik rétegen helyeztem el, és abba a pozícióba, ahol a mozgás



4. kép A kívánatos henger



5. kép Pontcsoportok létrehozása

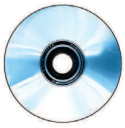
megkezdése előtt lennie kellett, egy másolatot helyeztem el róla. Amikor a tárgy mozgása elkezdődött, akkor az álló másolatot helyeztem át egy nem látható rétegre, a mozgó tárgyat pedig a látható rétegen jelenítettem meg. Ez az egyszerű példa jól megvilágítja, hogyan használható a ***Layer*** menüpont a kulcskockák létrehozásakor.
Ezek alapján, úgy gondolom, elég összetett animációkat készíthetünk a programmal, tehát a következőkben egy rövid bevezetés keretében ismerkedjünk meg a karakteranimáció alapjaival. Karakteranimáció készítése során valamilyen mozgó figurát készítünk – ő lesz a karakter –, majd ezt a figurát különféle mozdulatsorok elvégzésre tesszük képessé. Hamarosan csontalapú animációt hozunk létre, amiben a tárgy formáját a belsejében elhelyezett csontok mozgása fogja befolyásolni. Kezdetben

SHIFT-F6	Animációs görbék
ALT-A	Az animáció előzetes nézete
ALT-O	A kezdeti helyzetkülönbség eltávolítása
CTRL-P	Szülő–gyermek-viszony létrehozása
ALT-P	Szülő–gyermek-viszony megszüntetése
F7	Animációs beállítások (például útvonal követése)
SHIFT-S	Igazítás menü
CTRL-TAB	A csontváz helyzetének meghatározása

hozzunk létre egy hengert a ***Főmenü-Add-Mesh-Cylinder*** menüpontok kiválasztásával. Ezután kapcsoljunk abba a nézetbe, ahol a henger oldalnézetben látszik, és lépjünk be a pontszerkesztő módba. Jelöljük ki a henger egyik végén található pontokat, és az F9 billentyűvel váltsunk át a szerkesztő nézetbe. Itt alkalmazzuk az ***Extrude*** műveletet, és a kijelölt pontokat húzzuk el oldalirányba, majd szüntessük meg a kijelölést. Ezt követően jelöljük ki a henger másik végén lévő pontokat, és ismételten alkalmazzuk az előbbi kiterjesztést. Így, ha mindent jól ment, most a henger négy kört összekapcsoló síklapokból áll, ahogyan az a *4. képen* is látható.
Ezután lépjünk ki a pontszerkesztő módból, és hozzunk létre egy csontvázat, ami szintén látható a 4. képen.
A háromdimenziós kurzort állítsuk a henger egyik végéhez. Pontosabb igazítást érhetünk el, ha a hengert kijelöljük, pontszerkesztő módba váltunk át, és a henger egyik végén található középpontot kijelölve, a SHIFT-S billentyűk hatására megjelenő menüből a legalsó pontot kiválasztva a kurzort pontosan a kiválasztott ponthoz igazítjuk. Az előbb említett menü alkalmas különféle igazítások elvégzésére. A Blender a ***Sel->Grid*** hatására a kiválasztott objektumot a rácshoz igazítja, a ***Sel->Curs*** révén a tárgy a kurzorhoz igazodik, a ***Curs->Grid*** hatására pedig a kurzor igazodik a rácshoz, míg a ***Curs->Sel*** alkalmas arra, hogy a kurzort a tárgyhoz igazítsuk. Miután így meghatároztuk a csontváz kiindulópontját, hozzuk létre a mozgatást segítő csontokat. Alkalmazzuk a ***Főmenü-Add-Armature*** menüpontokat, majd az egyes csontokat helyezzük el a 4. képen látható helyeken.
Tudnunk kell, hogy a csontoknak kétféle szerkesztőmódját különböztetjük meg. Az egyik, amiben a csuklópontokat mozgathatjuk, a szokott módon, a TAB billentyűvel kapcsolhatjuk be, míg a másik módban a csontok helyzetét változtathatjuk. Ennek a módnak a neve a Blender szóhasználatában ***pose-mode***, vagyis helyzetmód, és a CTRL-TAB billentyűkkel érhető el. Miután pontosan elhelyeztük a pontokat, ideje lesz meghatározni, hogyan hassanak a tárgyra. Jelöljük ki a hengert, majd váltsunk pontszerkesztő módba. Itt úgynevezett pontcsoportokat kell létrehoznunk, amit az *5. képen* látható vezérlőelemekkel tehetünk meg. Az F9 billentyűvel valamelyik nézetet váltsuk át a szerkesztőmódba, majd keressük meg az 5. képen pirossal jelölt elemeket. Jelöljük ki a henger egyik végének a pontjait, és a képen látható ***New*** gombot használva hozzunk létre egy új pontcsoportot. Kereszteljük el „Bone” névre, vagyis a ***Weight*** mező feletti névmezőbe írjuk be ezt a szócskát. A kiválasztott az ***Assign*** gombbal pontokat rendeljük hozzá a csoporthoz, majd szüntessük meg a kijelölést. Ismételjük meg ezeket a műveleteket a többi hengerszeletre is, ám a továbbiakat ***Bone.001***, ***Bone.002***, ***Bone.003*** nevekkel illessük. Ezzel tulajdonképpen elkészültünk munkánk legbonyolultabb részével. Most már csupán azt kell a program tudtára adni, hogy ehhez a tárgyhoz mely csontok tartoznak. Kapcsoljunk tárgyszerkesztő módba, majd jelöljük ki először a hengert, azután a csontokat. A CTRL-P billentyűk hatására megjelenő menüből válasszuk ki a ***Use Armature*** pontot. Így a csontokat hozzárendeltük a tárgyhoz, ezután ha valamelyik csontot helyzetmódban elforgatjuk, a tárgy alakja követni fogja a változást. A hatás mértékét a pontcsoportok létrehozásánál látható ***Weight*** érték határozza meg.
A következő hónapban az inverz kinematikával fogunk foglalkozni, amiről előzetesen annyit érdemes tudni, hogy amikor ilyen megoldással mozgatunk egymáshoz kapcsolódó tárgyakat, akkor az összetett, egymáshoz kapcsolt részek közül elegendő csak az egyiket mozgatni, a többi tárgy pedig alkalmazkodik a megváltozott helyzethez. Élő példaként a saját lábunk szolgálhat. Képzeljük el, hogy olyan módon emeljük fel lábfejünket, hogy az egyik lábujjunkat megfogjuk. Ahogyan a lábfej elmozdul, a mozgást korlátozza és befolyásolja a lábszár és a comb hosszúsága és hajlékonysága. Egy másik példa lehet egy lánc lengése, amikor minden láncszem egyedileg járul hozzá a lánc mozgásához, és befolyásolja azt.
Végül az összefoglaló táblázatban láthatók az e részben használt és megjegyzésre érdemes billentyűkombinációk, a továbbiakban pedig sikeres animációkészítést kívánok minden kedves olvasómnak.

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

Smart



Napjaink merevlemezeinek a minősége hagy némi kívánnivalót maga után. Vajon előrejelezhető-e a meghibásodásuk, hogy megelőzhessük az adatvesztést?

A tapasztalatok azt mutatják, hogy sokkal nagyobb annak az esélye, hogy az újonnan vásárolt merevlemez az első 1–2 évben tönkremegy, mint korábban. (Elsősorban IDE-merevlemezekre gondolok, természetesen közöttük is akadnak kivételek.) Ennek oka lehet a fogyasztói társadalom szemlélete: már nem is éri meg minőségi termékeket előállítani, hiszen úgyis elavul. Esetleg a túl gyors technikai fejlődés és a rohamléptekkel haladó fejlesztések, amelyben a minőségbiztosítás már nem kap akkora szerepet, mint a hagyományos iparágakban. Még számos okot felsorolhatnánk, és mérlegelni lehetne, hogy melyik a jobb: olcsón rossz minőségű terméket vásárolni, vagy drágán jót, illetve hol a kettő között a határ. Nem az én feladatom ezt eldönteni, de annál inkább az, hogy megkeressem, mit lehet ellene tenni. Mivel ez a folyamat a tömeggyártás velejárója, és valószínűleg ez az irányzat továbbél, más oldalról kell megközelíteni a kérdést.

Az utóbbi években vásárolt merevlemezekben nagy eséllyel fellelhető a Smart (Self-Monitoring, Analysis and Reporting Technology), amely segíthet ebben. Hogy az adott típus támogatja-e, arról bővebb tájékoztatást a gyártó weblapján találhatunk, de már az 5–6 éve vett Quantum merevlemezemben is ott csücsül.

A merevlemezek meghibásodásának okait két részre lehet osztani: vannak előrejelezhető (predicted), és nem előrejelezhető hibák (non-predicted). Az utóbbiak közé tartoznak az ütődésből, túláramból fakadó hibák, ezek ellen odafigyeléssel és különböző berendezésekkel lehet védekezni. A megjósolható hibák közé tartozik például az olvasó/író fej repülőmagasságának a változása, a hibás írás, illetve az olvasás gyakorisága, a pozícionálási hibák, illetve bizonyos szempontból a hőmérséklet is. Ilyen jellegű hibákra a Smart is képes figyelni.

Működése leegyszerűsítve a következőképpen néz ki: az adott állapothoz egy értéket (egy egész számot) rendel hozzá. Ahogy romlik a merevlemez állapota, úgy nő ez az érték. Ha elér egy előre meghatározott határt, ami felett a meghibásodás valószínűsége már elég nagy, akkor érdemes lecserélni. Ilyen esetekben sem „szól" róla a merevlemez, nekünk kell folyamatosan figyelni (lekérdezni), hogy az adatvesztést elkerüljük. Természetesen ez a módszer sem képes minden meghibásodást teljes biztonsággal előrejelezni, ebből kifolyólag nem helyettesítheti a rendszeres biztonsági mentést sem.

A cikk további részében néhány Smartot használó programot mutatok be (`smart-suite`, `ide-smart` és a `hddtemp`), illetve azt, hogy miként is lehet őket munkára fogni. A programok leírása Debian GNU/Linux alatt készült, de csomagformában más nagyobb terjesztésekben is megtalálhatók, illetve forráskódból is feltehetők.

Mielőtt belevágnánk a programok telepítésébe, érdemes ellenőrizni, hogy a BIOS-ban engedélyezve van-e a Smart használata. Nem minden BIOS-ban található ilyen beállítási lehetőség.

ide-smart

Egy nagyon egyszerű, Smartot használó program. Az `ide-smart /dev/hda` parancs hatására az adott eszközről kilistázza az adatokat, ahogy az *1. listán* is látható. A kimenet értelmezése nagyon egyszerű: ha a `Threshold` eléri a `Value` értékét, az adott tulajdonság `Passed`-ról `Failed`-re vált, ekkor érdemes lecserélni a merevlemezt. A `Threshold` egy előre meghatározott érték, ami az adott típusra jellemző. A `Prefailure` közeli meghibásodást jelent, de ettől még akár évekig használni lehet. Az `Id` jelentése szintén típusfüggő, a `smart-suite` csomag ismer néhányat közülük. Táblázatunkban (50. CD Magazin/Smart könyvtára) a Smart által meghatározott `Id`-hez tartozó jelentések találhatók. A többi `Id` jelentése általában gyártónként különbözik. A *Kapcsolódó címek* között további érdekességeket találhatunk róla.

Ez a program kétféle tesztet ismer: online és offline. Az állandó elérésű próbákat a merevlemez folyamatosan, használat közben végzi el. Offline-próbát magunknak kell elindítani, például a következő paranccsal:

```
ide-smart -i /dev/hda
```

Ebben az esetben a próba azonnal elindul. Ha nem akarjuk rögtön elindítani, csak amikor az eszköz tétlen, akkor a következő parancsot adjuk ki:

```
ide-smart -1 /dev/hda
```

```
# ide-smart /dev/hda
Id=  1 Status=41 {Prefailure OffLine} Value=100 Threshold= 20 Passed
Id=  3 Status=39 {Prefailure Online } Value= 87 Threshold= 20 Passed
Id=  4 Status=50 {Advisory   Online } Value= 96 Threshold=  8 Passed
Id=  5 Status=51 {Prefailure Online } Value=100 Threshold= 20 Passed
Id=  7 Status=11 {Prefailure Online } Value=100 Threshold= 23 Passed
Id=  9 Status=18 {Advisory   Online } Value= 87 Threshold=  1 Passed
Id= 11 Status=19 {Prefailure Online } Value=100 Threshold= 20 Passed
Id= 12 Status=50 {Advisory   Online } Value= 96 Threshold=  8 Passed
Id= 13 Status=11 {Prefailure Online } Value=100 Threshold= 23 Passed
Id=199 Status=26 {Advisory   Online } Value=200 Threshold=  0 Passed
OffLineStatus=2 {Completed}, AutoOffLine=No, OffLineTimeout=0 minutes
OffLineCapability=3 {Immediate Auto SuspendOnCmd}
SmartRevision=6, CheckSum=189, SmartCapability=3 {SaveOnStandBy AutoSave}
```

```
# smartctl -v /dev/hda
Vendor Specific SMART Attributes with Thresholds:
Revision Number: 6
Attribute                        Flag       Value     Worst     Threshold     Raw Value
(  1)Raw Read Error Rate         0x0029     100       100       020           0
(  3)Spin Up Time                0x0027     087       087       020           5000
(  4)Start Stop Count            0x0032     096       09        008           2796
(  5)Reallocated Sector Ct       0x0033     100       100       020           0
(  7)Seek Error Rate             0x000b     100       100       023           0
(  9)Power On Hours              0x0012     087       087       001           8974
( 11)Calibration Retry Count     0x0013     100       100       020           0
( 12)Power Cycle Count           0x0032     096       096       008           2740
( 13)Read Soft Error Rate        0x000b     100       100       023           0
(199)UDMA CRC Error Count        0x001a     200       200       000           2
```

A súgóoldalakon további kapcsolókat is találni, de a fentiek voltak a leglényegesebbek.

smartsuite

Az előzőnél valamivel összetettebb programcsomag. Egy úgynevezett `smartd` démont tartalmaz, ami folyamatosan figyeli a merevlemezek (az IDE-t és SCSI-t egyaránt) állapotát, és a rendszernaplóba (syslog) bejegyzéseket készít. Futtatása egyszerűen a `smartd` paranccsal történik. A másik ebben a csomagban található program a `smartctl`. Szintén megkülönböztet állandó elérésű és offline-próbát, és az eddigi legrosszabb értéket is le tudja kérdezni. Beállításai a teljesség igénye nélkül:

- ***-c:*** ellenőrzi, hogy hibás-e a lemez.
- ***-v:*** az `ide-smart` kimenetéhez hasonló táblázatot listázza ki (lásd a 2. listát), azzal a különbséggel, hogy az egyes `Id`-k jelentését is ismeri, de nem hibátlanul. Például előfordulhat, hogy a bekapcsolt órák számának a tízszeresét írja ki.
- ***-e:*** engedélyezi a Smartot.
- ***-d:*** letiltja a Smartot.
- ***-O:*** azonnal elindít egy állandó elérésű próbát.
- ***-t:*** négyóránként offline-próbát indít;
- ***-a:*** minden adatot megjelenít, amit le tud kérdezni a Smarttól.

hddtemp

A csomag, Debian Sarge-tól érhető el. A programon kívül szükség lesz a ***hddtemp.db*** fájlra, amely adatokat tartalmaz arról, hogy a különböző típusú merevlemezek hogyan mutatják a hőmérsékletet. A ***hddtemp.db*** (50. CD Magazin/Smart könyvtárában is megtalálható), de Google-lal gyorsan és egyszerűen rá lehet találni. Alapértelmezés szerint az */etc* könyvtárban keresi, de megadható a `-f` kapcsolóval is. Futtatása nagyon egyszerű, például: `hddtemp /dev/hda`

A kimenet feldolgozása

Én a 4. listán (50. CD Magazin/Smart könyvtár) látható héjprogramot használom arra, hogy figyelje, ki kell-e cserélnem a merevlemezt. Nagyon egyszerű a működése: elektronikus levelet küld, ha egyes `threshold` értékek megközelítik a `value`-t, illetve ha `failed` tulajdonságot fedez fel. Megjegyzések alapján egyértelműen testreszabható. Szükséges hozzá egy működő MTA, `ide-smart` és `perl`. `cron` segítségével érdemes naponta vagy inkább óránként futtatni.
Ha valakinek ez nem elég, és figyelemmel szeretné kísérni, hogyan romlik a merevlemez állapota, MRTG-vel gyönyörű grafikonokat lehet róla gyártani. CD-mellékletünkön pár héjprogram megtalálható lesz hozzá, illetve az egyik korábbi írásomban (Linuxvilág 2002. november, 64. oldal) elmondottak alapján könnyen munkára is foghatók.

Tapasztalatok

Eddig egyszer fordult elő velem, hogy egy merevlemez különösebb előzmény nélkül hosszú idő után meghibásodott. Az egyik tulajdonság az `ide-smart` szerint `failed` lett. Lefuttattam rajta a `x` programot, de nem jelzett hibát. Másnap beszereztem egy új merevlemezt. A boltban azt mondták, hogy az nekik nem elég, ha a Smart szerint már hibás, csak akkor cserélik be, ha hibás szektor (badblock) is van rajta. Készítettem róla egy biztonsági mentést, majd futtattam rajta a ***read-write badblock*** próbát (`badblocks -vn`). Ez már több hibás szektort is jelzett. Ha tehát a Smart hibát jelez, általában csak néhány nap marad a meghibásodásig, így ilyenkor a lehető leghamarabb érdemes mindent menteni, és egy másik merevlemezt beszerezni.

Összefoglalás

Egy korábbi cikkemben írtam SW-RAID-ról, amely ugyan nem váltja ki a Smart használatát, de nagyon jól kombinálható vele. Ha például két RAID-1-be kötött merevlemez közül az egyik a Smart szerint meg fog hibásodni, akkor előre fel lehet készülni a leállásra.
Bár a merevlemezek minősége nem javul, ezzel a módszerrel a váratlan meghibásodásból származó adatvesztés valószínűsége mérsékelhető, és ennek köszönhetően sok bosszúságtól és idegeskedéstől kímélhetjük meg magunkat.

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.

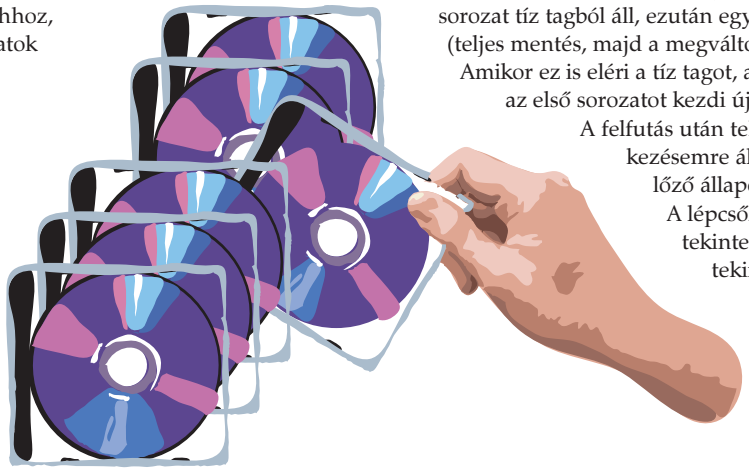
KAPCSOLÓDÓ CÍMEK

- http://www.santools.com/smartmon.html
- http://www.ucsc.edu/currents/99-00/09-27/smart.html
- http://www.siguardian.com/products/siguardian/on_line_help/about_s_m_a_r_t_technology.html
- http://www.unisysworld.com/monthly/2002/04/failure.shtml

Mentette már az e hetit?

Adatainkból egy példány nem példány. Két példány fél példány, három példány „a” példány, és négynél kezdődik a biztonság… Mindig büszke voltam rá, hogy az adataim – aránylag – biztonságban vannak, tanítványaimnak is nemegyszer beszéltem erről.

Mind az otthoni, mind a munkahelyi számítógépemen ugyanarra a munkakörnyezetre van szükségem: ezt egy fiókba (rack) tett kisebb merevlemez biztosítja, míg az operációs rendszer a ki nem vehető merevlemezen van, ami mellett azért marad még elég szabad hely ahhoz, hogy a fiókban lévő munkaadatok másodpéldánya elférjen.



Ablakos operációs rendszerű koromban az adatmentés igen egyszerűen történt: a gép kikapcsolása előtt egy kötegelt feldolgozás kitakarította a szemetet, majd a fiókban az úgynevezett archív bit felhasználásával a legutóbbi mentés óta megváltozott fájlokat igen könnyedén (egyetlen `xcopy` utasítással) tükröztem az operációs rendszer melletti szabad helyre.

Ezt a szokásomat a pingvin sem változtatta meg, egészen addig, amíg… amíg elég látványosan rá nem döbbentem arra, hogy egyszerű módszerem a hordozott merevlemez tönkremenetele (elvesztése) ellen ugyan megvéd, de a programban rejtőző hibák ellen nem sokat ér. Történt egy emlékezetes napon, hogy fájlrendszerhiba folytán a legutóbb használt két táblázatom keresztbe láncolódott – természetesen a kulcsfontosságú része veszett el, és az érdektelen megkétszereződött. Az üzemszerű mentés pedig megsemmisítette a kulcsfontosságú fájl előző másolatát… Amikor a jelenség megismétlődött, elhatároztam (mivel mégsem ütött meg a guta), hogy bonyolítom egy kissé az adatmentési módszeremet. Alapgondolatom nem új: lépcsőzetes mentésnek (incremental backup) hívják már jó ideje. Először a teljes munkaterület mentése történik meg egyetlen tar ***gzip*** fájlba, majd minden további alkalommal csak az időközben módosult állományokat mentem egy-egy újabb ***tgz***-be. Ilyen módon, ha bármilyen hibára fény derül, jó néhány mentett korábbi állapotom akad, és sokkal nagyobb valószínűséggel találok ép példányt egy fontos, de tönkrement állományból. Természetesen a rendelkezésemre álló hely véges, ezért egy ilyen sorozat tíz tagból áll, ezután egy új sorozat kezdődik (teljes mentés, majd a megváltozott állományok). Amikor ez is eléri a tíz tagot, a következő mentés az első sorozatot kezdi újra és így tovább.

A felfutás után tehát bármikor rendelkezésemre áll legalább tíz megelőző állapot, legfeljebb húsz. A lépcsőzetességre való tekintettel a helyigény sem tekinthető túlzottnak.

Természetesen a mellékelt programocska nem általános csodaszer. Első és talán legfontosabb korlátja, hogy a mentendő fájlok listáját a legutóbbi módosítás dátuma alapján állítja össze. A PC-k belső naptárára viszont semmilyen garancia nincs… A lépcsőzetesség és a valószínűleg felhasználható hely miatt alapvetően irodai, illetve programozói jellegű géphasználat esetén alkalmazható, ahol nem túl sok és nem túl nagy fájl keletkezik vagy módosul (ellentétben mondjuk a videovágással és -szerkesztéssel).

A mentés időigénye teljes mentésnél (a sorozatok első tagja) pár perc, a továbbiakban ennek töredéke. Ennyit szerintem megér a lényegesen megnövelt adatbiztonság.

Keszthelyi András (KEA@turul.banki.hu)

```
#!/bin/bash
#
# Az alanti 4 változó értéke egyéni ízlés
# szerint állítható
MaxNoOfFiles=10         # Legalacsonyabb
                        # értéke: 2
MunkaKtar="/mnt/hordd"  # Nincs / a
                        # könyvtárnevek végén!
MMP="/mnt/hda4"         # Mentes Mount Pont
MKT="ment"              # Mentes KonyvTar

##############################################

CKT=$MMP/$MKT           # célkönyvtár
Ser1="Ment1_"
Ser2="Ment2_"
Teljes="Nem"

# fájlrendszer-befűzések ellenőrzése devfs
# esetén a cutot másképpen kell paraméterezni
if [ qwq`mount | grep $MunkaKtar |
➥cut -d " " -f 3` != qwq$MunkaKtar ]; then
   echo "A munkakönyvtár nem érhető el,
         ➥mentés nem történt"
   exit
fi
```

folytatás a következő oldalon

folytatás az előző oldalról

```
# csak a ktar elérhetőséget kell vizsgálnunk,
# vö. lentebb az első futással!
if [ ! -d $CKT ]; then
   echo "A célkönyvtár nem érhető el, mentés
        ➥nem történt"
   exit
fi

# szabad hely ellenőrzése
let Kell=`df | grep $MunkaKtar |
➥tr -s " " | cut -d " " -f 3`
let  Van=`df | grep $MMP | tr -s " " |
➥cut -d " " -f 4`
if [ $Van -le $Kell ]; then
   echo "Nem biztos, hogy elegendő a hely
        ➥a mentéshez, nem csináljuk"
   exit
fi

# kitakarítjuk a szemetet kiterjesztés alapján
# válogatunk, egyéni ízlés szerint bővíthető
echo -n "Takaritas... "
find $MunkaKtar -type f \( -name \*.bak -o
➥-name \*.$\* -o -name \*.img -o -name \*.tmp
➥-o -name \*.wbk \) -exec rm {} \;
echo "vege"

#Hány sorozat és mennyi darab van:
if [ `ls $CKT | wc -l` -eq 0 ]; then
   echo "Ures kt., elso futas..."
   Teljes="Igen"
   touch -t 198001010000 ${CKT}/mentdatum.txt
   echo "${Ser1}000-000000" >
        ➥${CKT}/lastfile.txt
fi

FC1=`ls ${CKT}/${Ser1}* 2> /dev/null | wc -l`
FC2=`ls ${CKT}/${Ser2}* 2> /dev/null | wc -l`

SerNo=`cat ${CKT}/lastfile.txt | cut -c 5`

if [ $SerNo -eq 1 ]; then
   #Legutóbbi mentés az 1. jelű sorozatba ment
   FNamePrefix=$CKT/${Ser1}
   if [ $FC1 -ge $MaxNoOfFiles ]; then
      # Megvan a sorozatból a max. darabszám,
      # meg kell kezdeni a 2. jelű sorozatot,
      # (ha van régebbről, törlendő)
      FNamePrefix=$CKT/${Ser2}
      rm $CKT/${Ser2}*
      find $CKT -type f -name ${Ser2}\*
           ➥-maxdepth 1 -exec rm {} \;
      echo "2. sorozat torolve: `date`"
           ➥>> ${CKT}/mentdatum.txt
      Teljes="Igen"
   fi
fi

if [ $SerNo -eq 2 ]; then
   #Legutóbbi mentés a 2. jelű sorozatba ment
   FNamePrefix=$CKT/${Ser2}
   if [ $FC2 -ge $MaxNoOfFiles ]; then
      # Megvan a sorozatból a max. darabszám,
      # meg kell kezdeni az 1. jelű sorozatot
      # (ha van régebbről, törlendő)
      FNamePrefix=$CKT/${Ser1}
      find $CKT -type f -name ${Ser1}\*
      -maxdepth 1 -exec rm {} \;
      echo "1. sorozat torolve: `date`"
           ➥>> ${CKT}/mentdatum.txt
      Teljes="Igen"
   fi
fi

# Ezen a ponton: ha az egyik sorozatunk
# elérte a max. darabszámot, letöröltük a
# másikat és beállítottuk a fájlnév elejét,
# meg hogy teljes v. részleges mentést kell
# csinálni. Itt jön maga a mentés.
echo -n "A tenyleges mentes kov.: "


FName=$FNamePrefix$(date +%j)-$
➥(date +%H%M%S).tgz

if [ "$Teljes" = "Igen" ]; then
   tar -czf $FName $MunkaKtar 2 > /dev/null
   echo -n "teljes mentes... "
else
   # csak a frissebbeket kell menteni
   echo -n "frissebbek mentese... "
   find $MunkaKtar -type f -anewer
   ➥${CKT}/mentdatum.txt -exec ls {} \;
   ➥> /tmp/mentlista.$$
   tar -czf $FName -T /tmp/mentlista.$$
   ➥2 > /dev/null
   if [ -f /tmp/mentlista.$$ ];
   ➥then rm /tmp/mentlista.$$; fi
fi
basename `echo $FName` > ${CKT}/lastfile.txt
basename `echo $FName` >> ${CKT}/mentdatum.txt
echo "rendben(?)."

###EOF
```


Héjprogramozás Linux alatt (4. rész)

Sorozatunk előző részeiben megismerkedhettünk a változókkal, a logikai kifejezésekkel, a döntési szerkezetekkel, a parancsbehelyettesítés módszerével és még néhány más, a héjprogramozás szempontjából alapvető lehetőséggel.

Ebben a részben szintén a héj alapszolgáltatásairól, a ciklusok szervezéséről és a függvényekről lesz szó. Minden programban, így a héjprogramokban is gyakran fordulnak elő ismetlődő részek. Ha ezekre a program különböző pontjain van szükség, akkor elkülönített blokkokban, függvényekben helyezhetjük el őket, ha viszont egy helyen kell ugyanazt a műveletsort sokszor végrehajtani, akkor ciklust szervezünk. Erről a két lehetőségről lesz szó ebben a részben.

Új feladat

Tegyük fel, hogy létezik egy több sorból és több oszlopból álló, kizárólag egész értékeket tartalmazó számtáblázatunk, amihez elő szeretnénk állítani a sorok, illetve oszlopok összegét. Írjunk olyan héjprogramot, ami elvégzi az összegzést, ha a táblázatot tartalmazó fájl nevét megadjuk, és a szabályos kimeneten megjeleníti az eredményt. Továbbá egy parancssori kapcsolóval azt is szabályozni lehessen, hogy a sorok vagy az oszlopok összegzését kell-e elvégeznie.
Azonnal érezzük, hogy itt tulajdonképpen két faladat megoldásáról van szó, vagyis a kész program kétféle teendőt lesz képes ellátni. Ennek megfelelően a kódot is célszerű lesz olyan módon kialakítani, hogy a két műveletet megvalósító részek jól elkülönüljenek egymástól. A magas szintű programnyelvekben ezt a moduláris felépítést eljárások vagy függvények segítségével valósítják meg. Természetesen héjprogramokban is lehetőségünk nyílik függvények megadására.

Függvények

A függvény a teljes program elkülönült, jól meghatározott feladatot ellátó része. Lehetnek paraméterei és van kimenete. Amolyan állam az államban. Héjprogramok esetében egy függvény általános alakja a következő:

```
függvénynév()
{
   ... a függvény törzse ...
}
```

Ha az így elkülönített programblokkot futtatni akarjuk, elegendő a függvény nevét beírni a főprogram (vagy egy másik függvény) megfelelő pontján. Lényeges, hogy a gömbölyű zárójeleket héjprogramok esetében ***nem*** kell kiírni. Ezek a függvény megadását és nem a hívását jelzik a parancsértelmező számára.

Ciklusok

Ahhoz, hogy egy táblázat sorain vagy oszlopain végig tudjunk haladni, nyilván ciklusokra lesz szükségünk. Héjprogramokban alapvetően kétféle ciklus létezik. Az egyiknél a más nyelvekben is szokásos módon egy logikai feltételt kell megadnunk, és a ciklus addig fut, ameddig ez a feltétel igaz. Ennek a szerkezetnek az alakja a következő:

```
while logikai_kifejezés
do
   ... ciklusmag ...
done
```

A logikai kifejezés megfogalmazására ugyanazok a szabályok érvényesek, mint amiket az előző részben a `test` parancs kapcsán már bemutattunk.
A másik ciklustípusnál meg kell adnunk azt a sorozatot, aminek az értékeit a ciklusváltozónak fel kell vennie. Általános alakja a következő:

```
for ciklusváltozó in lista
do
   ... ciklusmag ...
done
```

Ez természetesen nem feltétlenül jelenti azt, hogy a listát kézzel kell begépelnünk. Megadhatunk a helyén egy olyan héjváltozót is, ami a megfelelő sorozatot tartalmazza, magát a sorozatot pedig más utasításokkal is előállíthatjuk.

A sorok összegzése

Ennyi bevezető után írjuk meg a program lelkét képező két függvényt. Kezdjük a sorokon belül összegző algoritmus megírásával (lásd az *1. listát*).
A 3. sorban egyszerűen megszámoljuk, hogy hány sorból áll a táblázat.
Látható, hogy mindkét ciklustípusra szükségünk volt.

```
 1: sorosszegzes()
 2: {
 3:    sorszam=`cat $1 | wc -l`
 4:    i=1
 5:    while [ $i -le $sorszam ]
 6:    do
 7:       egysor=`head -$i $1 | tail -1`
 8:       osszeg=0
 9:       for elem in $egysor
10:       do
11:          osszeg=`expr $osszeg + $elem`
12:       done
13:       echo $osszeg
14:       i=`expr $i + 1`
15:    done
16: }
```

```
 1: oszloposszegzes()
 2: {
 3:   sorszam=`cat $1 | wc -l`
 4:   oszlopszam=`head -1 $1 | wc -w`
 5:   i=1
 6:   while [ $i -le $oszlopszam ]
 7:   do
 8:     osszeg=0
 9:     j=1
10:     while [ $j -le $sorszam ]
11:     do
12:       egysor=`head -$j $1 | tail -1`
13:       k=1
14:       for elem in $egysor
15:       do
16:         if [ $k -eq $i ]
17:         then
18:           osszeg=`expr $osszeg + $elem`
19:         fi
20:         k=`expr $k + 1`
21:       done
22:       j=`expr $j + 1`
23:     done
24:     echo $osszeg
25:     i=`expr $i + 1`
26:   done
27: }
```

Az 5. sorban induló `while` ciklus a bemeneti fájl sorain halad végig, és a 7. sorban mindig a megadott sorszámút vágja ki. A `head` parancs egy fájl elejéről jeleníti meg a megadott számú sort, a `tail` pedig ugyanezt teszi, de a fájl végéről visszafelé haladva. A 7. sorban előbb kivágjuk a bemenet első i sorát, majd ezekből csak az utolsót vesszük. Az eredmény egyetlen sor, mégpedig pontosan az i-edik.
A 9. sorban induló `for` ciklus ezen az egyetlen soron (`egysor`) mint listán halad végig, és az `osszeg` nevű változóban gyűjti a részeredményeket. Az összegzést az `expr` parancs segítségével végezzük, ami egy kizárólag egész számokkal működő „számológép”. Azt is érdemes megjegyezni, hogy a műveleti jelet a tulajdonságoktól (argumentum) mindig szóköznek kell elválasztania, ha pedig szorozni akarunk vele, akkor a szorzásjelet `\*` alakban kell megadni, a csillag karakter különleges jelentése miatt.

Az oszlopok összegzése

Az oszlopok összegzése ennél egy kissé bonyolultabb, bár a megvalósítás hasonló az előzőhöz. Itt három ciklusra van szükségünk. Az első (6. sor) a feldolgozott oszlopokat számolja, a második (10. sor) a bemenet sorain halad végig, a harmadik (14. sor) pedig minden sorból csak a pillanatnyi oszlopindexnek ($i) megfelelő sorszámú tagot adja hozzá a részösszeghez (lásd a 2. *listát*).
Valószínűleg az olvasó is érzi, hogy ez nem túl elegáns megoldás, sőt talán már arra is rájött, hogy a feladat lényege az oszlopok címzése. Nincs olyan egyszerű módszerünk, amivel ki tudnánk emelni egy sorozatnak mondjuk a harmadik elemét. (Valójában létezik ilyen módszer, de ez már egy másik történet lesz.)
Figyeljük meg, hogy a ciklusváltozók növelését szintén az `expr` segítségével oldottuk meg.

```
 1: if [ $# -ne 2 ]
 2: then
 3:   echo "A program használata:"
 4:   echo $0 "[-s | -o] fájlnév"
 5:   exit 1
 6: fi
 7:
 8: if [ ! -f $2 ]
 9: then
10:   echo "A megadott fájl nem létezik!"
11:   exit 2
12: fi
13:
14: case $1 in
15: "-s") sorosszegzes $2;;
16: "-o") oszloposszegzes $2;;
17: *) echo "Ismeretlen kapcsoló!" ; exit 3
;;
18: esac*
```

A főprogram

Ezzel a feladatot lényegében meg is oldottuk. Az utolsó lépés a vezérlést ellátó főprogram megírása, és a teljes kód összeállítása. A feladat kiírása szerint a sorok és oszlopok összegzése között egy kapcsolóval lehet váltani. Ez azt jelenti, hogy a programunknak két parancssori kapcsolója lesz. Az első a kapcsoló, a második pedig a feldolgozandó fájl neve (3. *lista*).
A főprogram elején megvizsgáljuk, hogy valóban két kapcsolót kaptunk-e a parancssorban (1. sor), majd a biztonság kedvéért azt is leellenőrizzük, hogy a megadott fájl létezik-e. Hibás kapcsolószám esetén a unixos szokásoknak megfelelően tájékoztató szöveget íratunk ki. Figyeljük meg, hogy a különböző hibákat eltérő visszatérési értékkel jelezzük.
A 14. sorban egy az első parancssori kapcsolóban (`$1`) megadott érték alapján működő többszörös elágazás kezdődik. Esetünkben itt dől el, hogy a két függvény közül melyik fog lefutni. A lehetőségeket karakterláncok formájában kell megadnunk (15. és 16. sor), a hozzájuk tartozó műveletsorok végét pedig kettős pontosvessző jelzi. A * (csillag) karakter a „bármi más” jelölésére szolgál. Esetünkben ez csak ismeretlen kapcsoló lehet, így hibaüzenetet küldünk rá. Magát a `case` szerkezetet az `esac` (case visszafelé) kulcsszó zárja le.
Figyeljük meg, hogy a függvények meghívásánál azok parancssori kapcsolójaként `$2`-t (a feldolgozandó fájl nevét) adjuk meg, maguk a függvények azonban `$1`-ként hivatkoznak erre. Ez azért van így, mert a függvényeknek önálló értékkészletük van, aminek így a számozása sem azonos a főprograméval.
A változókkal kicsit bonyolultabb a helyzet. A függvény belelát a főprogram változóiba, és ugyanez fordítva is igaz (röviden tehát minden változó globális). Ugyanakkor ha a függvény hoz létre egy változót, akkor a főprogram csak az első függvényhívás után tud rá hivatkozni.
A teljes program a fenti három részlet egyszerű összemásolásával előállítható.

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

Könyvtárkezelés Perlben

Általánosan elfogadott nézet, hogy amit C-ben nem lehet megoldani, azt más nyelvben sem. Mára újabb egyetemes tétellel gazdagodtunk: amit Perlben nem lehet egyszerűen megoldani, azt nem lehet más nyelvben sem.

A fenti állítást bizonyítandó nézzük meg közelebbről a könyvtárkezelés Perl-beli megvalósítását. Mint látni fogod, semmivel sem bonyolultabb könyvtárlistát készíteni, mint állományt kiíratni. Szemügyre vesszük a könyvtármutatókat, szó lesz egy objektumközpontú felületről, továbbá a jogosultságok és a tulajdonos beállításáról.

Az állománytípusok

Kezdjük az elején! Egy könyvtárlista elemei nem egységesen fájlok, hanem típusuk szerint különbözhetnek. A közvetett hivatkozástól (symbolic link) a könyvtáron át egészen a hálózati foglalatig számos fajtát határozhatunk meg. A Perlben minden típusnak megfelelően létezik egy műveleti jel (operator), ami igazat ad vissza, ha az elem az adott típushoz tartozik, ellenkező esetben hamisat. Hasonló ez a bashből ismert módszerhez, a két nyelv műveleti jelei között vannak azonban eltérések. Az alábbi egyszerű függvény egy skalárt ad vissza, ami a megadott állomány típusát jellemzi.

```
sub fileType {
     my $file = shift;
     return "szimbolikus link" if -l $file;
     return "konyvtar" if -d $file;
     return "fifo" if -p $file;
     return "karakter-specialis" if -c $file;
     return "blokk-specialis" if -b $file;
     return "socket" if -S $file;
     return "szoveg" if -T $file;
     return "binaris" if -B $file;
     return "ismeretlen";
}
```

Így egyszerűen meghatározhatod egy állomány típusát, a legkönnyebben az alábbi szemléletes sorral foghatod munkára az eljárást:

```
print fileType "/dev/hda";
```

Természetesen a nyelv rugalmasságából eredően állománynév helyett fájlmutatót is megadhattunk volna, most azonban az egyszerűség kedvéért a könnyebb utat választottam.

A könyvtármutatók

Mint azt az elején említettem, könyvtárlistát gyerekjáték kiíratni. Nem elég, hogy nem nehezebb egy állomány tartalmának a megjelenítésénél, de egy az egyben ugyanazt az eljárást igényli. A szemmel látható különbség mindössze a függvények neveiben jelentkezik. A végletekig leegyszerűsítve azt mondanám, hogy csupán minden, az állománykezelésnél megismert függvény neve mögé egy `dir`-t kell ragasztani. Az `opendir` függvény segítségével megnyitjuk a könyvtárat, a későbbiekben az értékként átadott könyvtármutatóval fogunk hivatkozni rá. A gyémánt műveleti jel (< >) megfelelője ebben az esetben a `readdir`. Végül a `closedir` segítségével töröljük a könyvtármutatót a névtérből. Lássuk mindezt a gyakorlatban!

```
my ($dir,$file,$type);
($dir = $ARGV[0]) or die "Hasznalat: ".$0."
                         ➥<konyvtar>\n";
opendir (DIR, $dir) or die $dir.
       ➥": ".$!."\n";
print $dir." tartalma:\n";
while ($file = readdir(DIR)) {
     $type = fileType($dir."/".$file);
     print "\t".$file." (".$type.")\n";
}
closedir (DIR);
```

A fenti program a korábban már tárgyalt `fileType` függvényt használja az állomány típusának a meghatározásához, és egy egészen tetszetős kimenetet hoz létre. Mint láthatod, nincs különbség az állománykezeléshez képest, illetve csak igen csekély. Egy könyvtármutatót nem lehet írási műveletre felhasználni. Továbbá a `readdir` nem használja az alapértelmezett változót ($_), így egyet kötelező megadni.
Érdemes megjegyezni, hogy a `readdir` a gyémánt műveleti jelhez hasonlóan tömb és skalár összefüggésben is használható. Ez azt jelenti, hogy a kiíratás úgy is megoldható, hogy egyetlen `readdir` hívással egy `@files` tömbbe az összes bejegyzést begyűjtöd, majd egy `for $file (@files) { }` szerkezettel végigléptetted az elemeken. Az előbbi ciklusban minden egyes lefutásnál a `$file` a következő értéket veszi fel.
Ne felejtsd el, hogy a könyvtárlista tartalmazni fogja a `'.'` és a `'..'` (a pont és a két pont) elemeket is. Amennyiben ezekre nem vagy kíváncsi, a ciklusban egy mintaillesztéssel szűrd ki őket:

```
next if $file =~ /^\.\.?$/;
```

A DirHandle modul

Az alapkönyvtár része az említett modul, ami objektumközpontú felületként szolgál a már megismert alacsonyabb szintű függvényekhez. Ennek értelmében a `DirHandle` használata nem jelent több szolgáltatást, mindössze más megközelítése ugyanannak a feladatnak. Lássunk egy példát!

```
use DirHandle;

my ($dir,$dh,$file,$type);
($dir = $ARGV[0]) or die "Hasznalat: ".$0."
                         ➥<konyvtar>\n";
($dh = new DirHandle ($dir)) or die $dir.
       ➥": ".$!."\n";
```

```
print $dir." tartalma:\n";
while ($file = $dh->read) {
    next if $file =~ /^\.\.?$/;
    $type = fileType($dir."/".$file);
    print "\t".$file." (".$type.")\n";
}
$dh->close;
```

Pillanatnyi könyvtár

Egy parancsértelmezőhöz hasonlóan a Perl is ismeri a pillanatnyi könyvtár fogalmát. Ez alapértelmezésben az a könyvtár, ahonnan a parancsfájlt indítottad. Nem feltétlenül egyezik meg azzal a hellyel, ahol a program található. Erre különös tekintettel figyelj! A legjobb, ha ahol csak lehet, abszolút elérési utat használsz.
A jelenlegi munkakönyvtárad lekérdezhető a `Cwd` modul részét képező `cwd` függvénnyel. A `chdir` pedig – ahogy a nevéből is sejtheted – a pillanatnyi könyvtár megváltoztatására használható. Ezek egy bashből kiadott `pwd`, illetve `cd` parancshoz hasonlóan viselkednek.

A umask függvény

Egy állomány létrehozása esetén a jogosultságokat tekintve a Perl eredeténél fogva Unix-szerűen viselkedik. Létezik egy alapértelmezett maszk, amit az új állomány jogainak meghatározásához használ fel. Ezt hívják `umask`-nak. Röviden szólva ez egy olyan nyolcas számrendszerbeli szám, ami megmutatja a megvont jogokat. Ez az érték jellemzően 022. Ez azt jelenti, hogy minden engedélyezett, leszámítva a „csoport” és a „többiek” írási jogát. Így az újonnan létrehozott állományok jogosultsága 644, a könyvtáraké 755.
A Perl `umask` függvénye nyújt lehetőséget az említett alapértelmezett maszk megváltoztatására. Azt például, hogy az újonnan létrehozott állományokra minden engedélyezve legyen, az alábbi függvényhívással érheted el:

```
umask 0000;
```

A chmod függvény

A `chmod` segítségével explicit módon is megadhatod egy állomány vagy könyvtár elérési jogait. A függvény értékként egy listát vár, aminek az első eleme az új jogosultság nyolcas számrendszerben megadva, a további elemek pedig állományok. Ha gondban vagy a nyolcas számrendszerrel, az `Fcntl` modullal számok helyett beszédesebb állandókat használhatsz. Íme a példa:

```
use Fcntl ':mode';
chmod S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH|S_IXOTH,
"alkalmazas.pl";
```

A fenti állandók értelmezése:

```
S_IRWXU = S_I Read Write eXecute User
➥(a tulajdonosnak olvasás, írás, futtatás)
S_IRGRP = S_I Read GRouP
➥(a csoportnak olvasás)
S_IXGRP = S_I eXecute GRouP
➥(a csoportnak futtatás)
S_IROTH = S_I Read OTHer
➥(a többieknek olvasás)
S_IXOTH = S_I eXecute OTHer
➥(a többieknek futtatás)
```

Ezeket bitenként egy VAGY művelettel összegeztük, így ugyanazt kaptuk, mintha 0755-öt adtunk volna meg.

A chown függvény

A `chown` segítségével megváltoztathatod egy, esetleg több könyvtár vagy állomány tulajdonosát, illetve csoportját.
A függvény a `chmod`-hoz hasonlóan egy listát vár. Ennek első eleme az új felhasználói azonosító, a második a csoportazonosító, a további értékek állománynevek. Természetesen lehetőség nyílik csak az egyik, illetve a másik megváltoztatására, ekkor annak az azonosítónak a helyén, amit nem kívánunk módosítani, `-1`-et kell megadnunk.
Kényelmetlen azonban számokkal dolgozni – a Perl itt két újabb függvénnyel áll a rendelkezésünkre. A `getpwnam`, illetve a `getgrent` a */etc/passwd*-ből és a */etc/group*-ból nyerik ki a névhez tartozó azonosítót. Az alábbi példa a ***szoveg.txt*** tulajdonosát `balazs`-ra változtatja:

```
chown getpwnam("balazs"), "-1", "szoveg.txt";
```

Az mkdir, az rmdir és az unlink

A bámulatosan beszédes nevű függvények pont azt teszik, amit a parancsértelmező által futtatott programok is tennének.
Az `mkdir` új könyvtárat hoz létre; kötelező érték a létrehozandó könyvtár neve, ezután a jogosultság is megadható. Az utóbbi elhagyása esetén a `umask`-nak megfelelően jön létre a könyvtár.

```
mkdir "appz", 0755;
```

Az `rmdir` egy könyvtárat töröl. Értékként a könyvtár neve szerepel. Fontos, hogy a rendszerből ismert testvéréhez hasonlóan csak akkor szüntet meg egy könyvtárat, ha az üres.

```
rmdir "appz";
```

Az `unlink` állományokat töröl. Értékként a törlendő állományok neveivel egy listát vár.

```
unlink "alma.txt", "korte.txt", "szilva.txt";
```

Miért foglalkoztunk ezekkel?

Miért érdemes ezzel foglalkozni, ha ott van a Midnight Commander? Azért, mert mindennapos feladat lehet egy könyvtár alatti összes könyvtárnak adott jogosultságúra, míg az összes állománynak más jogosultságúra történő beállítása. Ezt kézzel tíz bejegyzés esetén még az őrület elkerülésével meg lehet oldani, ezer esetén már aligha. Sokszor eszedbe juthat, hogy erre programot írni felesleges, gyorsabban végzel, ha kézzel csinálod. Szerintem ez súlyos tévedés. Egyrészt legközelebb is ütközhetsz ilyen feladatba, másrészt a tapasztalat azt mutatja, hogy hamarabb végzel egy parancsfájllal, mint a nyílbillentyűkkel az mc-ben. Arról nem is beszélve, hogy a számítógépet pontosan erre találták ki: a feladatok önműködővé tételére.
Kellemes programozást!

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

A Zope CMF

Működtessük weblapunk munkafolyamat-rendszerét tartalomszolgáltatóink és felhasználóink megelégedésére. Fejlesszünk Python alapú tartalomkezelő keretrendszerrel!

Az elmúlt néhány hónapban megtudhattunk néhány dolgot a tartalomkezelésről, illetve egy kicsit részletesebben is megismerkedhettünk a nyílt forrású Plone tartalomkezelő rendszerrel (CMS). A Plone igen rátermett CMS rendszer, olyasmi, amit az ember telepítés után majdhogynem azonnal használni tud. A Plone legnagyobb előnye az egyszerűség. Könnyű telepíteni, könnyű használni és egyszerű beállítani.

A Plone (vagy bármely más CMS) testreszabásának azonban megvannak a határai. Ha mint nem kereskedelmi termék nem tetszik a Plone kinézete, természetesen megváltoztathatjuk egy másik modellre. De ha teljesen más munkafolyamatot szeretnénk, mint amit a Plone nyújt, valószínűleg csak az időnket vesztegetjük, ha ilyesfajta változtatást próbálunk megvalósítani. Sokkal több értelme van, ha megírjuk saját CMS rendszerünket, vagy (ha bonyolultabb a feladat) átszabjuk azokat, amelyeket rugalmasan építettek fel.

A legtöbb CMS-terjesztő mostanra már ráébredt, hogyha használható terméket szeretnének, annak testreszabhatónak kell lennie. Amennyiben J2EE alapú CMS rendszert vásárolunk (márpedig a kereskedelmi CMS rendszerek igen nagy százaléka a J2EE-n alapul), várhatóan lehetőségünk lesz a tartalomtípusokat és azok kezelési módját leíró, új Java-objektumokat készíteni. Bizonyos ponton túl elmosódik a határvonal a meglévő CMS testreszabása és a saját CMS rendszer létrehozása között.

Itt lép be a képbe a Zope tartalomkezelője (CMF), amit olyan módon terveztek meg, hogy hatékony hátteret nyújtson saját CMS-kezelőink megírásához. Mivel a Zope-fejlesztések igen gyorsak és egyszerűek, és mivel a Zope meglévő háttéreszközeit is felhasználhatjuk, könnyen elképzelhető, hogy legalább ugyanolyan gyorsan (viszont sokkal olcsóbban) ki tudjuk fejleszteni CMS rendszerünket, mint ahogyan azt valamilyen kereskedelmi CMS-megvalósításban megtehetnénk. A Plone nem önálló program, hanem a CMF-re épül. Ennek megfelelően, ahogy a CMF fejlődik, úgy a Plone is egyre jobbá válik.

Ebben a hónapban a CMF lesz terítéken, ami a Zope-alkalmazások közül lassan a figyelem középpontjába kerül. A CMF jelenleg elérhető legutóbbi üzembiztos változata az 1.3.1.-es, az alfaváltozat pedig a 2.0-s számot viseli. Ha esetleg az elkövetkezendő években a Zope irányvonala még nem lenne teljesen tiszta, a CMF 2.0 egyértelműen „a Zope3 némileg újracsomagolt vezető alkalmazása”.

A CMF telepítése és beállítása

A CMF telepítését néhány hónappal ezelőtt már bemutattuk (lásd a Kovácsműhely rovatot a Linuxvilág 2003. májusi számában), ügyhogy most nem részletezem.

Töltsük le a forrásfájlokat a ➲ http://cmf.zope.org lapról, majd csomagoljuk ki a Zope saját könyvtárunk alatti ***lib/python/Products*** könyvtárába.

Ezt követően a CMF könyvtárból a ***Product*** könyvtár ***CMFCore, CMFDefault***, ***CMFTopic*** és ***CMFCalendar*** fájljaira készítsünk közvetett hivatkozásokat. Indítsuk újra a Zope-ot, és a Zope kezelőablakában az ***Add*** menü alatt máris rengeteg új CMF-vonatkozású terméket fogunk látni.

Mielőtt bármilyen CMF-objektumot hozhatnánk létre, előbb CMF-oldalunkhoz kell készítenünk egy tárolót. Nyilván észrevettük az egyértelmű párhuzamot a Plone-oldal és a CMF-oldal létrehozása között. Az új CMF-oldal létrehozásához a webalapú Zope kezelőfelületben egyszerűen csak válasszuk az ***Add*** menü ***CMF Site*** pontját. Ezt követően CMF-oldalunkat el kell neveznünk, illetve meg kell adnunk a leírását.

A CMF-oldal létrehozásakor válaszolnunk kell arra a kérdésre, hogy új felhasználói könyvtárat szeretnénk-e létrehozni az adott CMF-oldalon belül, vagy egy már létező felhasználói könyvtárat szeretnénk felhasználni. Egyelőre tegyük az utóbbit, ami azt jelenti, hogy a legfelsőbb szintű Zope-oldalon megadott felhasználók lesznek egyúttal CMF-oldalunk felhasználói is. Amennyiben CMF-oldalunkat inkább önálló egységként szeretnénk létrehozni, és (a lap tulajdonosán kívül) nem szeretnénk a külső világra hivatkozni, nyugodtan létrehozhatunk saját felhasználói könyvtárakat is.

Miután a CMF-oldal létrehozásával végeztünk, a honlapra kerülünk, ahol megtudhatjuk, hogy ideje meglátogatnunk az alapbeállítások űrlapját. Minthogy a CMF eredetileg Portal Toolkit néven (PTK) volt ismert, sok lap CMF-oldalak helyett portálokat emleget. Az űrlapba írandó adatok meglehetősen általános természetűek. Itt állíthatjuk be például azt a levélcímet, amiről (látszólag) a létrejövő elektronikus levelek érkeznek, valamint az oldal SMTP-kiszolgálóját.

A dolgok mindjárt érdekesebbé válnak, ha az útmutatást követve átlépünk a CMF-kezelőfelületre, ami valójában a CMF-oldalhoz tartozó Zope-kezelőfelület. Más szavakkal, ha oldalunk a ***/cmfdemo*** címen érhető el, az oldal tartalmába a ***/cmfdemo/manage*** címen nézhetünk bele. A kezelőfelület, mint azt már Zope alatt megszokhattuk, a bal oldalon egy kis navigációs menüt tartalmaz. Ugyanakkor, mint azt a múlt hónapban láthattuk, a bal oldalon számos portáleszközt is találunk, amelyekkel a CMF-oldalunkat állíthatjuk be és módosíthatjuk.

A ***portal catalog***-ra kattintva megjelenik a szótár (angolul „vocabulary”, a Zope világában ezt a szót használják az index helyett), amiből megtudhatjuk, hogy az oldal gazdája miképpen készíthet teljes körű szövegkeresést a CMF-oldalon – minden erőfeszítés nélkül. A ***portal types***-ra (portáltípusok) kattintva a tartalom objektumosztályokat pillanthatjuk meg. Ezek az osztályok alkotják a CMF magját. Alább részletesebben is megvizsgáljuk a tartalomosztályokat, a jövő

hónapban pedig azt is megtudhatjuk, hogyan készíthetjük el a saját tartalomtípusainkat.
Végül kattintsunk a ***portal_workflow***-ra, ahol az egyes tartalomtípusokhoz használni kívánt ***workflow*** (munkafolyamat) objektum címét gépelhetjük be. A munkafolyamat szabályozza, hogyan jut el a tartalom az írástól a kiadásig, illetve biztosítja, hogy az emberek bizonyos feladatok elvégzéséhez a megfelelő jogosultságokkal rendelkezzenek. A szerzők például történeteket írhatnak, de nem tehetik közzé írásaikat az oldalon.
A jó munkafolyamat-rendszer lehetővé teszi, hogy a saját szervezeti igényeinknek megfelelően magunk állítsuk fel ezeket a szabályokat.

Egyszerű CMF-lap

A CMF-vezérlőpanel bemutatása után ideje megvizsgálnunk a lapunkat is. Ha nyers CMF-oldalunkra belépünk, középen a fő tartalomterületet, továbbá különféle helyeken eszköztárakat és dobozokat láthatunk. A legfölső menüben találjuk meg a fontosabb navigációs pontokat, amikkel az oldal elejére, a tagok oldalára, a hírek oldalára, illetve az oldal tartalmának a keresésére ugorhatunk. A menü alatt, de még mindig a jobb felső sarokban felhasználói menükre bukkanhatunk, első helyen a ***My Preferences*** (saját beállítások) ponttal. Itt a bejelentkezett felhasználók felvehetik a saját beállításaikat, hivatkozásokat adhatnak kedvenceik listájához, illetve kijelentkezhetnek. A még be nem jelentkezett felhasználóknak a rendszer felkínálja a bejelentkezést, amennyiben létezik azonosítójuk, illetve akiknek még nincsen, lehetőséget kapnak a csatlakozásra. Bal oldalon találjuk a navigációs menüt, ebben láthatjuk az elérhető könyvtárak listáját, illetve beállíthatunk néhány képességet, például a hírügynökséget (syndication) és a helyi szabályokat.
Aki már hozzászokott a Plone-oldalakhoz, az az alapértelmezett CMF-oldalt egy kicsit talán spártainak, de mindenképpen ismerősnek fogja találni. Az egyszerűség oka, hogy az alapértelmezett CMF-oldal arra készült, hogy egy saját CMS rendszerben felhasználhassuk; igaz, hogy teljes mértékben működőképes, de nem önálló alkalmazásnak szánták. Új tartalomtípusok meghatározásával és a megjelenítő felületek módosításával pillanatok alatt életre kelthetjük a CMS-t. Mivel a megjelenítő logika a rendszer többi részétől teljesen elkülönül, viszonylag gyorsan megváltoztathatjuk a kinézetet.
A CMF-oldal minden egyes tagjához egy vagy több szerepet rendelhetünk: Member (tag), Reviewer (bíráló), Manager (vezető) vagy Owner (tulajdonos). Ezek a bíráló kivételével valószínűleg ismerősen csengenek a tapasztalt Zope-felhasználók és rendszerfelügyelők fülének. Az új szerep a munkafolyamatok kezeléséhez szükséges, ahol a bírálók feladata hitelesíteni a tartalmat, még mielőtt az felkerülhetne a Webre.
A rendszerfelügyelők a képernyő bal oldalán kiterjedt menüt láthatnak, megtekinthetik a pillanatnyi könyvtár tartalomnézetét, illetve módosíthatják a meglévő tartalmat, valamint új tartalmat hozhatnak létre. Amikor új objektumot hozunk létre, nemcsak tartalmat és azonosítót (ID) rendelünk hozzá (amit a hagyományos webrendszerek fájlnévnek neveznek), hanem egyúttal a tartalmat leíró metaadatokat is. Bár az objektum típusát létrehozás után már nem változtathatjuk meg, valamennyi paraméterét átírhatjuk, ha a tartalomnézetbe visszatérve megnyitjuk a kérdéses tartalmat.
Minden egyes tartalmat ki kell adni (publish), mielőtt azt mások is láthatnák. Alapértelmezés szerint az új tartalom állapota ***private*** (magán), de a bal oldali menüsor ***publish*** (kiadás) hivatkozásával kiadhatjuk őket. Ugyanezt a felületet használva a lap felügyelője cikkeket vonhat vissza a lap kiadott elemeinek listájából. Ez igen nagy fejlődés a hagyományos weblapokhoz képest, ahol el kellett távolítanunk a csatolásokat és le kellett törölnünk a fájlokat.
Továbbá a legtöbb tartalomtípus vitafórummal rendelkezhet. Ez a ***Comment on This Posting*** (Megjegyzés ehhez az íráshoz) képességhez hasonlít, ami igen népszerű a weblogok világában, mivel lehetővé teszi, hogy a látogató megjegyzéseket fűzzön az oldal hivatalos rendszerfelügyelője által feltett írásokhoz. Amikor egy új tartalmat felrakunk, eldönthetjük, hogy elfogadjuk-e a vitafórumokra vonatkozó alapértelmezés szerinti oldalmeghatározást, vagy ennél az egy példánynál felülbíráljuk a hely globális beállításait.

Tartalomtípusok

Nézzük meg, hogy tulajdonképpen milyen tartalomtípusokat is hozhatunk létre.
A legtöbbjük a CMFDefault termékben van meghatározva, a ***lib/python/Products/CMFDefault*** könyvtár különálló ***.py***-állományaiban. Ez a termék írja le a Zope kezelőfelületen korábban látott beállítóeszközöket, valamint az olyan CMF alatt példányosítható alap-tartalomtípusokat, mint a ***NewsItem*** (hírelem), a ***Portal***, az ***Image*** (kép) és a ***Link*** (hivatkozás).
A hozzám hasonlókat minden bizonnyal meglepi és lenyűgözi a CMF-ben meghatározott alap-tartalomtípusok egészen kis mérete. A lista alján a ***NewsItem*** áll, mindössze száz sorral, de még a ***Portal*** elem is csak 350 sort számlál. Ez nemcsak azt jelenti, hogy ezeket a tartalomtípusokat szükség esetén könnyű megváltoztatni és a hibáikat felderíteni, hanem azt is, hogy az új tartalomtípusok felvétele is igen egyszerű. Igazság szerint mára már számos új tartalomtípus született a CMF-hez, és továbbiak is várhatóak. Ha ellátogatunk a CMF Collective Projekt honlapjára (➲ http://collective.sf.net), számtalan CMF-vonatkozású terméket találhatunk, amik az elmúlt hónapokban jelentek meg. Születőfélben lévő CMF-termékek találhatók többek között e-kereskedelem, fotóalbum és weblog témakörben. Ahogy a CMF egyre népszerűbbé válik, várhatóan a CMF Collective népszerűsége is egyaránt növekedni fog.

Összefoglalás

Figyelembe véve, hogy a Zope Corp. többször kijelentette, hogy a Zope jövője a CMF, és tudván, hogy a CMS telepítése felháborítóan költséges is lehet, egyértelmű, hogy a Zope Corp. komolyan megpróbálja túlszárnyalni kereskedelmi vetélytársait, és azok árai alá ígér. Ugyanakkor, mivel a Zope és a CMF nyílt forrású, saját termékeinkben nyugodtan felhasználhatjuk, akár a tartalomkezelésről szeretnénk többet megtudni, akár különböző elemeket szeretnénk szerkeszteni és kiadni.
A következő hónapban egy kicsit mélyebben elmerülünk a témában, és megnézzük, hogyan írhatjuk meg a saját CMF példa-tartalomtípusunkat.

Linux Journal 2003. július, 111. 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'in-ben él.

Alkalmazzunk XML-t!

Manapság rengeteget lehet hallani az XML-ről. Egyes programok reklámját olvasva néha már-már forradalmi újdonságnak tűnhet. De mi is valójában, és mire használható?

Anélkül, hogy részletekbe bocsátkoznánk: az XML a HTML-hez hasonlóan az SGML-re épülő leíró nyelv. Amiben leginkább eltér tőle, az az, hogy nincsenek egy meghatározott feladatra (mint például a HTML-nél weboldalak leírására) szolgáló tagok előre meghatározva, hanem tetszőlegesen bővíthető, lényegében csak a formai (syntactic) szabályok adottak. Ebből eredően rendkívül rugalmas, mindenre ráhúzható, és vélhetően ennek köszönhető nagy népszerűsége is.
Ezzel együtt ha valaki belenéz egy-egy XML-t tartalmazó fájlba, észreveheti, hogy a szigorú szabályok miatt rendkívül redundáns, „bőbeszédű". Sok esetben az adattartalom szinte elvész a formai elemek között. Felmerül a kérdés, hogy akkor miért használják olyan sokszor, mik az előnyei?

- Jól körülírt, szabványjellegű leírással bír, ha valaki tartja magát ehhez, akkor az általa készített XML-fájlt később és más programmal is el lehet olvasni.
- Mivel a HTML-hez hasonlóan nem bináris kódokat tartalmaz, szükség esetén ember által is olvasható, illetve írható.
- Nagyon jó elemzőkönyvtárak készültek hozzá, amiket mindenféle programozási nyelvekhez illesztettek, így viszonylag könnyen és gyorsan lehet olyan programot írni, ami egy XML-állományt be tud olvasni, emellett az előállítása is egyszerű.
- Mivel a HTML (némi megszorításokkal) az XML részhalmazának tekinthető, lehetséges a kettő kombinálása (ahogy ez a korszerű böngészőkben részben meg is történik).
- A beépített feldolgozási utasítások (process instructions, PIs) és névterek (name spaces) segítségével, valamint a külső stíluslapok felhasználásával szinte programnyelv-szerű, egészen bonyolult felhasználása is lehetséges.
- Tetszőlegesen bővíthető új elemekkel (tagokkal).

Néhány XML-alkalmazási területet is megemlítenék:

- OpenOffice adatcsere-formátumként (elég nagyok a fájlok) alkalmazzák.
- A .NET és a .GNU rendszerek szintén adatátvitelre használják.
- A fentiek régebbi és egyszerűbb változata az XMLRPC távoli függvényhívásoknál a függvények változóinak és az eredmény átvitelére is alkalmas.
- Számos program a beállítások tárolására használja.
- Nos, igen, az én HTML/PHP-előállító programom bemeneteként is ezt választottam.

Az utóbbi lesz írásom témája.

A program

Néhány évvel ezelőtt készítenem kellett néhány honlapot, de gyorsan belefáradtam, hogy állandóan ugyanazokat a HTML-tagokat gépeljem be (a vizuális szerkesztők nem tetszettek, mivel sok tekintetben korlátoznak), ezért először néhány osztályt elkészítettem PHP-ben – ezek már sok mindenben segítettek. Hamarosan azonban néhány nehézség ütötte fel a fejét: a PHP-ben nehéz igazi osztályokat készíteni, még nehezebb (legalábbis az akkori, 3.0-s változatban) mutatókat alkalmazni. Ráadásul nem sikerült megvalósítanom benne a fő célomat: olyan elemeket létrehozni, amiket utána ugyanúgy használhatok, mintha a nyelv (HTML) részei lennének.
A fentiek miatt elhatároztam, hogy egy másik programot írok. Mivel akkoriban ódzkodtam a C-től, a Perl nyelvet választottam. Igaz, hogy lassúbb, viszont a programtesztelés gyorsabb, és szinte minden könyvtár illesztését megoldották hozzá (lásd CPAN). Fő elvárásaim a következők voltak:

- Legyen „kézzel", ember által írható bemeneti formátuma.
- HTML/PHP-kimenet és önműködő előállítás jellemezze.
- A Forth nyelv szavaihoz vagy a C függvényeihez hasonlóan az egyszer már meghatározott elemek a régiekhez hasonlóan, teljes értékűen, intuitív módon használhatóak legyenek. Ezt hiányoltam a leginkább a legtöbb HTML-előállító programból.
- A HTML-tagok minden meghatározás nélkül is működjenek (azaz egy HTML-fájl apró módosításokkal, átlátszóan kerüljön át rajta. Minimális módosítása az XML szigorúbb alaki szabályai miatt szükséges, de erről később bővebben szólok).

Célkitűzéseim részben meg is valósultak. Később más feladataim adódtak, és nem foglalkoztam a HTML-lel és a weboldalkészítéssel, így egy kissé elhanyagoltam a témát. Most közzéteszem, hátha valakinek ötletet ad, esetleg kedve támad továbbfejleszteni.
Röviden a program igényeiről: én a Perl 5.05-ös változattal írtam. Azóta vannak újabbak, elvileg futnia kell velük. Kisebb nehézségeket esetleg okozhat, hogy az újabb Perl-változatok mindenáron Unicode-ban szeretnének dolgozni. Szükség esetén a kimeneten alkalmazható egy szűrő, hogy a kívánt kódkészletben kapjuk meg az eredményt (például iso-latin-2). Az `XML::Parser` modulra is szükség lesz, az általam írt `XML::Element` modulhoz. (Figyelem, később jöttem rá, hogy ilyen nevű modul valamelyik CPAN-os csomagban már létezik. Ha azt telepítettük, akkor az enyémet nevezzük vagy helyezzük át, így kerülve el az ütközést.) Amennyiben adatbázis-kötést is használni kívánunk, akkor a PostgreSQL `Pg` modulját is szükségünk lesz. Ha képátalakításokra (conversion) is szeretnénk használni, az `Image::Magick` modul is kell.
Ha ezek megvannak, akkor a programot (xtend) másoljuk valahová a burkunk elérési útjába (központi telepítésnél, több felhasználó számára */usr/bin* vagy */usr/local/bin*)), az előre létrehozott elemeimet pedig az xtend program `$global` változójában megadott könyvtárba (alapban */usr/local/httpd/xml_mod*, de ez megváltoztatható). Az ***Element.pm*** modult a ***Parser.pm***-mel azonos könyvtárban helyezzük el.

Példa az index.xml-re

```
<!DOCTYPE GENERAL [
<!ENTITY atl "gr/transp.gif">
]>
<!--  `rnyékot vető lap. Egypontos konténer.  rtékek:
      arnyek                : az árnyék szélessége
      lapszin               : lap színe
      arnyekszin            : árnyék színe
      width                 : teljes szélesség
      lalign                : lap align
      align, valign, height  : TD attribútumok a tartalomra vonatkoznak
      -->
<_def arnyek="6" lapszin="white" lalign='center' arnyekszin="gray" width="100%"
                ➥align="center" valign="center" height="">
<table align='!lalign' width="!width" cellspacing="0 " cellpadding="0 " border="0 ">
<tr><td width="!arnyek" height="!arnyek" bgcolor="!lapszin">
<img src="&atl;" width="!arnyek" height="!arnyek"/></td>
<td width="100%" height="!arnyek" bgcolor="!lapszin">
<img src="&atl;" height="!arnyek"/></td>
<td width="!arnyek" height="!arnyek">
<img src="&atl;" width="!arnyek" height="!arnyek"/></td></tr>
<tr><td width="!arnyek" bgcolor="!lapszin">
<img src="&atl;"/></td>
<td width="100%" bgcolor="!lapszin" align="!align" valign="!valign" height="!height">
<__>Ide jön a body...</__>
</td>
<td width="!arnyek" bgcolor="!arnyekszin">
<img src="&atl;"/></td></tr>
<tr><td width="!arnyek" height="!arnyek">
<img src="&atl;" width="!arnyek" height="!arnyek"/></td>
<td width="100%" height="!arnyek" bgcolor="!arnyekszin">
<img src="&atl;" height="!arnyek"/></td>
<td width="!arnyek" height="!arnyek" bgcolor="!arnyekszin">
<img src="&atl;" width="!arnyek" height="!arnyek"/></td></tr>
</table>
</_def>
```

Röviden a működésről

A program bemenetként XML formátumú fájlt vár. Mint később látni fogjuk, a működés egy XML ***MakeFile*** használata segítségével jelentős mértékben bővíthető. Célszerű is így használni, mivel néhány képesség másként nem érhető el (szándékosan írom így, nagy betűvel kezdve a szavakat a névben, hogy megkülönböztessem az „igazi” `makefile`-tól). Mivel az XML elég érzékeny a formai szabályok betartására (valójában a legkisebb hiba esetén elszáll), felhívnám a figyelmet az XML néhány sajátosságára:

- A tagoknak nyitó- és záróféllel is rendelkezniük kell. Például:
  `<valami> ... </valami>`
- Tartalom nélküli tagokat is lehet használni, de akkor önmagában kell zártnak lennie. Például a HTML img tagját így kell használni:
  `<img src='kep.jpg' />`
- A tagok minden tulajdonságát (attributum) idézőjel vagy egyszeres idézőjel (aposztróf) között kell megadni, a HTML-lel ellentétben akkor is, ha egy szóból áll.
- A teljes fájlnak is egy nyitó, illetve záró tag között kell lennie, ha más nincs, erre jó a `<html></html>`
- A formai szabályok leírásánál használt karakterek nem szerepelhetnek a szöveges törzsekben, mert ettől az elemző kiakad. Ilyenkor használjuk a `<!CDATA[[` és `]]>` határolókat, az ezek közötti részt nem elemzi.
- Megjegyzést a HTML-ben megszokott `<!--` és `-->` határolók közé írhatunk.
- Készíthetünk egységeket (entity), amiket az `&nev;` formában utóbb a tagok tulajdonságaiban és a szöveges törzsben használhatunk. A meghatározás formája:
  `<!DOCTYPE GENERAL [ <!ENTITY nev "meghatározás"> ]>`
  Ennek a fájl elején kell lennie, az első valódi tag előtt. Természetesen több egységet is alkothatunk, ekkor több `ENTITY` következik egymás után.

Az egyszer már meghatározott elemek újrafelhasználhatóságát olyan módon oldottam meg, hogy minden meghatározást külön fájlba helyeztem, ezek a tag nevét viselik, ***.xml*** kiterjesztéssel. Annak érdekében, hogy különböző szintű meghatározásaink lehessenek, a program három rétegű keresést alkalmaz: először az adott ***.xml***-fájlt keresi abban a könyvtárban, amelyből futtatjuk, azaz ahol a fő fájl helyezkedik el;

utána az úgynevezett ***site*** könyvtárban, majd a globális könyvtárban (lásd fent is). Ha megtalálja, akkor önhívó (recursive) módon kezdi el elemezni őket. Ha nem találja meg, akkor az adott tagot egy az egyben átmásolja a kimenetre: így megoldott, hogy az eredeti HTML-elemek általunk írt meghatározás nélkül is használhatóak legyenek.
Nos, mivel ez eddig elég száraz volt, lássunk egy eleven példát! Tegyük fel, hogy kialakítunk egy (elsőre egyszerű) HTML-oldalt, amit több lap alapjául szeretnénk használni. Ennek meghatározása az ***alap.xml*** fájlban a következő legyen:

```
<_def tag='alap' szin='white'>
<html>
<body bgcolor='!szin'>
 <__/>
</body>
</html>
</_def>
```

Itt két különleges tagot láthatunk: az `_def` és a `___` nevűt. Általában a „rendszer” tagokat aláhúzásjellel kezdem (ez más programnyelvekben is szokásos), a megkülönböztetés érdekében. Az `_def` amellett, hogy biztosítja az XML-nek szükséges bennfoglaló tagot, alapértelmezett tulajdonságokat tud felvenni. Ebben a példában ilyen a `szin`, amelynek értéke `white`. A másik tag az `___` nevű. Ennek az a feladata, hogy a tároló típusú tag meghatározásában kijelölje azt a helyet, ahol a felhasználáskor megadott tartalom bemásolásra kerül. Lássuk, hogyan használjuk ezt, és mindjárt érthetőbb lesz! Az ***index.xml*** fájlba írjuk be a következőket:

```
<alap>
Ide jön a tartalom.
</alap>
```

Ezután a következőképpen „fordítsuk le”:

```
xtend index.xml > index.html
```

Ekkor a következő tartalmú ***index.html*** fájlt kapjuk:

```
<html>
<body bgcolor='white'>
Ide jön a tartalom.
</body>
</html>
```

Mint látható, a meghatározásnak megfelelően kibővítette a bemenetet (ezért neveztem xtend-nek, azaz extend-nek a programot). Mivel `szin` tulajdonságot nem adtunk meg az alaptagnak, az alapértelmezettet használta. Ha most ezt írnánk:

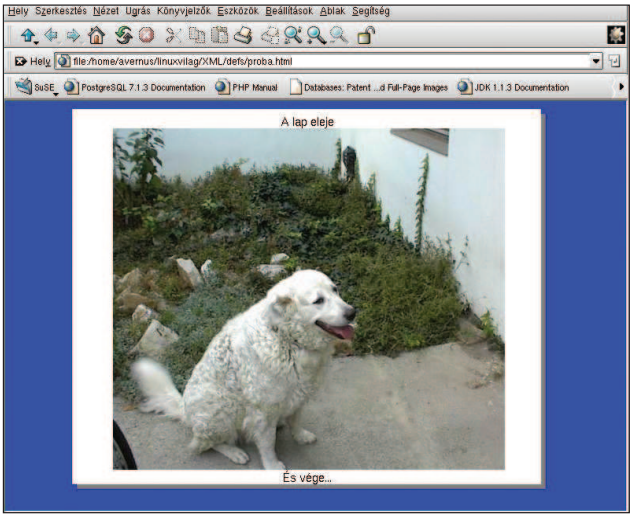
```
<alap szin='yellow'>
```

akkor a `bgcolor` értéke is ez lenne, azaz az alapértelmezettet felülírhatjuk.
Mielőtt valaki kijelentené, hogy ez eléggé fejletlen dolog, nézzünk meg az első részben egy bonyolultabb példát is. Kidolgoztam egy olyan keretet, ami úgy néz ki, mintha (megfelelő színválasztás esetén) egy papírlapon lenne a szöveg vagy más tartalom, és árnyékot vetne a háttérre. A neve legyen: lap1. Meghatározása a rendszermeghatározások között található meg, lásd *listánkon* (előző oldal).

A rendszermeghatározások közötti különbséghez képest annyit módosítottam, hogy fent az egység meghatározásában elvettem a bevezető / (per)jelet. Ez akkor szükséges, ha az oldalt HTTP-kiszolgálón keresztül kérjük le, és a képeket a törzskönyvtárból induló ***gr*** könyvtárban helyezzük el, nem pedig az adott fájl könyvtára alatt.
Ezután a következő bemeneti fájl segítségével próbáljuk is ki:

```
<html>
<body bgcolor='blue'>
<lap1 width='80%'>
A lap eleje <br/>
<img src='gr/kutyus.jpg' />
<br/>
 s vége...
</lap1>
</body>
</html>
```

Ezt lefordítva a *képünkön* látható eredményhez jutunk.



Mint látható, egy meglehetősen összetett HTML-fájlt kaptunk, a bemenet viszont kifejezetten egyszerű. Természetesen azt is megtehetjük, hogy a lap leírásából még több részt viszünk át az alapmeghatározásba: ebben az esetben az is lehetséges, hogy az egyedi fájlokban most csak a `<lap1>` és `</lap1>` közötti részeket írjuk be.
Sorozatunk következő részében megnézzük, hogyan tudjuk az XML ***MakeFile*** segítségével egy teljes honlap oldalszerkezetét leírni, milyen módon segít ez a különböző navigációs menük elkészítésében, valamint további elemekkel ismerkedünk meg. A későbbiek folyamán sorra kerül a program mint képkezelő (átméretezés, vágás, montázskészítés stb.) alkalmazása, valamint az SQL-illesztés, amivel nagyon elegánsan készíthetünk például árlistákat. Észrevételeket a hf@hmvhely.hu címre várok.

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

Új, idegen nyelvek felfedezése

Keressük meg a megfelelő szót, tanuljuk meg kiejteni, és a kívánt szöveget fordítsuk le magyar nyelvre.

Ez az egész egymás megértéséről szól, François. Valahányszor párbeszédbe elegyedünk valakivel, aki egy másik országból jött, arra teszünk próbát, hogy közös alapot találjunk az eszmecseréhez, általában egy olyan nyelv választásával, amelyet mindketten értünk, vagy ha ez nem megy, akkor oda-visszafordítgatással.

Quoi? Természetesen igazad van, mon ami. Néha nem egyszerű dolog magunkat megértetni, még azokkal sem, akik ugyanazt a nyelvet beszélik. Hiába gondoljuk, hogy ismerjük egy bizonyos szó jelentését, semmi nem biztosítja, hogy akihez beszélünk, a szót ugyanolyan módon értelmezi.

Ez az egyik oka az értelmező szótárak létezésének – és a Scrabble létezésének is.

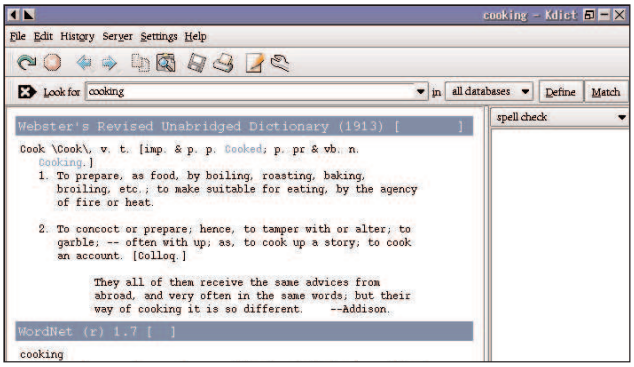
Á, mes amis! Örülök, hogy látlak titeket! Isten hozott titeket Chez Marcelnél, az ízletes Linux-konyha és a világ legkitűnőbb borainak otthonában! Helyezzétek magatokat kényelembe! François és én éppen arról beszélgettünk, hogy milyen kihívást jelent megértetni magunkat másokkal, és a valódi jelentést eljuttatni valakihez. François, neked mindez már nem új, úgyhogy gyorsan szaladj le a pincébe, és hozd vissza azt az 1999-es Napa Valley Cabernet Sauvignont, amit korábban kóstolgattunk – vagyis inkább minőségi ellenőrzésnek vetettünk alá.

A szavak nagyon fontosak, a megfelelő szavak pedig még inkább, mint azt bármelyik író el tudná nekünk mondani. Ez különösen akkor van így, amikor olyasvalakivel próbáljuk felvenni a kapcsolatot, aki nem beszéli a nyelvünket. Linux-rendszerünkkel élvezetes órákat tölthetünk annak megismerésével, hogyan javíthatjuk ezt a megértést.

Előfordulhat, hogy a szavak jelentése csak egérkattintásnyira van tőlünk. Ha a KDE 3.0-s vagy későbbi változatát futtatjuk, kipróbálhatjuk a következő fogást. Tegyük fel, hogy a cooking szó értelmezését keressük. Nyissuk ki a Konquerort, gépeljük be a cím helyére a ***dict: cooking*** szöveget, és nyomjunk ENTER-t. A Konqueror ennek hatására keresést hajt végre a Merriam-Webster hálózatos szótárában. Ha a szinonimaszótárban szeretnénk ugyanezt a keresést végrehajtani, a ***ths: cooking*** szöveget kell az előbbi helyett beírnunk.

A KDE a kdenetwork csomag részeként Kdict néven egy ragyogó beépített szótárprogramot is tartalmaz. A legvalószínűbb, hogy a programot a KDE futtatómenü (a nagy K) ***Utilities*** (segédprogramok) menüpontja alatt találjuk. A programot nevének begépelésével parancshéjból is futtathatjuk (lásd az *1. képet*). Lépjünk be, és a Kdict csatlakozik a különböző hálózaton keresztül elérhető szótárakhoz, hogy előkeresse az alkalmas meghatározást. A források között szerepel többek között a Merriam–Webster szótár, a Wordnet, a Jargon File és a Devil's Dictionary.

A Kdict villámgyors eléréséhez Kicker panelünkre feltehetünk egy kézre álló kisalkalmazást. Ennek módja a következő: jobb kattintás a nagy K betűn, a ***Panel*** menü →***Add***→***Applet***→***Dictionary*** kiválasztása. Most egy új program-kisalkalmazást kell látnunk Dictionary felirattal és három kis gombbal a Kicker panel jobb



1. kép A Kdict könnyű szótári keresést biztosít a hálózaton keresztül

2. kép A Kdict azonnali elérése

felső részén. Első indításkor csak a C gomb látható (a kiválasztott szöveg megadása), a másik kettő szürke.

Gépeljünk be egy szöveget a kis ablakba – akár csak egy szót vagy kifejezést –, nyomjunk ENTER-t, a Kdict pedig megjeleníti azokat a meghatározásokat, amiket a különféle forrásokból össze sikerült gyűjtenie. Arra is lehetőségünk van, hogy az éppen nézett weboldalon vagy dokumentumban kijelöljünk egy szót (kettős kattintás), és a kisalkalmazás C gombjára kattintsunk. A Kdict önműködően elindul és megszerzi a kiválasztott szó meghatározását.

Ha nem a KDE-t használjuk, vagy ha az egyszerűbb megoldásokat részesítjük előnyben, hadd mutassak be egy pehelykönnyű szöveges módú ügyfélprogramot, ami hasonló szolgáltatást nyújt. Ez Vishal Verma edict nevű programja – kíváncsi vagyok, hogy kikerestétek-e a szó jelentését a szótárból.

A programot az ➲ http://edictionary.sourceforge.net címről tölthetjük le. Az `edict` nem több, mint egy Perl-parancsfájl, de a feladatát így is rendesen elvégzi. A telepítésről szólva tényleg nincs sokkal több teendőnk, mint kicsomagolni a tarral és gzippel becsomagolt állományokat. A parancsfájlt abból a könyvtárból futtathatjuk, ahová kicsomagoltuk, de több mint valószínű, hogy inkább a `make install` paranccsal akarunk a ***/usr/bin*** könyvtárba a fájlról mentést készíteni.

A program futtatásához írjuk be az `edict` parancsot, majd mögé azt a szót, aminek a jelentését meg szeretnénk vele kerestetni. Szinonimakereséshez az `ethes` parancs mögé kell a keresett szót beírni. Ha az általunk keresett szót nem találja, egyéb választási lehetőségeket ajánl.

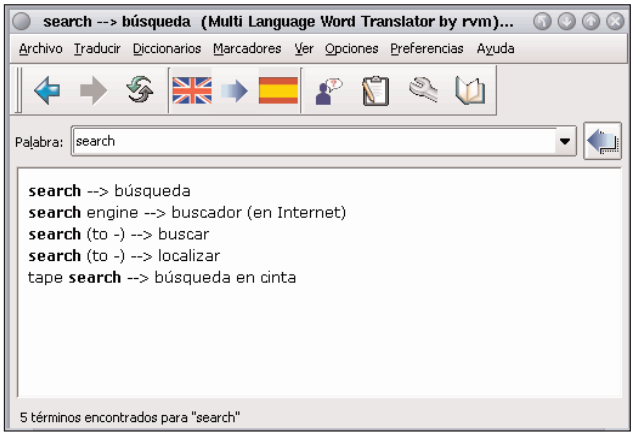
Az `ethes` program egy egyszerű közvetett hivatkozás, ami az `edict`-re mutat, ebből következően a szinonimakeresés lényegében ugyanazt a folyamatot indítja el, csak az eredmény lesz különböző:

```
[marcel@mysystem edict]$ ethes program
edict - Your personal command line dictionary.
Verison 1.0.
Looking up "program" in Merriam Webster Online
Thesaurus...
Entry Word: program
Function: noun
Text: 1 a formulated plan listing things
➥to be done
or to take place especially in chronological
➥order
<the program of a concert>
Synonyms: agenda, calendar, card, docket,
programma, schedule, sked, timetable
Related Words: bill; slate; plan
Idioms order of the day 2
Synonyms: COURSE 3, line, policy, polity,
➥procedure
```

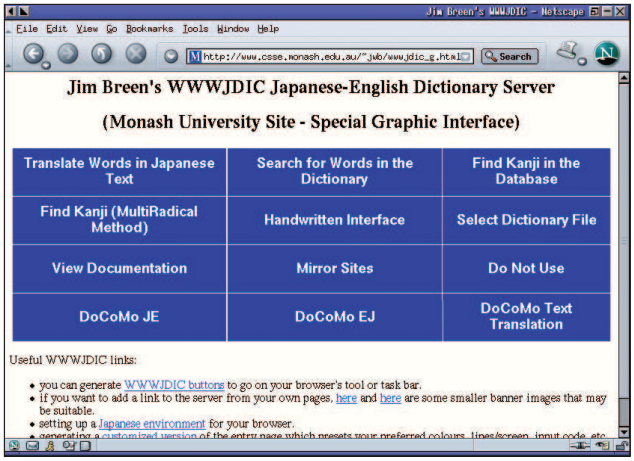
Ha már a szótárakról beszélünk, mi a helyzet akkor, amikor maga a szótár szólal meg? Vagy még tényszerűbben: mit érdemes elmondani ***Jeffrey Clement*** MWSpeaker nevű programjáról, amit úgy jellemez, mint „a valaha létezett legrosszabb beszédszintetizáló program”? Ezek az ő szavai, nem az enyémek. A gondolat egyszerű, de lehet, hogy egy kicsit bolondos. Beírunk egy szót vagy kifejezést, az MWSpeaker pedig emberi hangon felolvassa nekünk a beírtakat. A kérdéses beszéd a Merriam–Webster hálózati szótárából származik. Röviden, a program megkeresi az egyes szavakhoz tartozó wav-fájlt, letölti, és sorrendben lejátssza őket.
Lévén az MWSpeaker egy Python-parancsfájl, igazából még fordításra sincs szükség. Egyszerűen töltsük le a programot a ➲ http://www.jclement.ca/Projects/mwspeaker címről, ezután pedig bontsuk ki a tar és gzip által tömörített csomagot. Mielőtt ténylegesen használatba vennénk a programot, szükségünk lesz még néhány további csomagra, leginkább a wxPythonra, a pygame-re és a PythonCardra. A program futtatásához lépjünk be az ***mwspeaker-1.0*** könyvtárba (ahová az MWSpeakert kibontottuk), és gépeljük be a következő sorokat:

```
mkdir data
python mwspeaker.pyw
```

A ***data*** lesz az a könyvtár, ahol a wav-fájlok tárolása történik. A felhasználói felület nagyon egyszerű. Írjunk be egy szót vagy kifejezést, kattintsunk a ***Say it*** feliratra, és várjunk. Azért mondom, hogy várjunk, mert az MWSpeaker minden egyes szónak szép sorjában letölti a wav-fájlját, még mielőtt lejátszaná a szöveget. Az eredmény nagyon szórakoztató, mert a kiválasztott szavakat előadó hangok igen különbözőek lehetnek – előfordulhat, hogy a férfi és női hangok egy furcsa keverékét kapjuk eredményül. Mindez nagyszerű az angol nyelv szempontjából, de a Linux és a nyílt forrású programozók végül is a világ minden táján megtalálhatók, ahogy a Linux-felhasználók is. Igaz, hogy egy angol értelmező szótár hasznos azoknak, akik az angolt nem az első nyelvükként tartják számon, de néha fordításra is szükség van. François, remplisser les verres de nos invités, s'il vous plâit. Néha úgy érezzük, hogy ahhoz, hogy a franciát (vagy az olaszt, a spanyolt, a németet) angol nyelvre le tudjuk fordítani, szükségünk van egy Bábel-halra. De mi az a Bábel-hal? – kérdezhetitek. ***Douglas Adams***, a Galaxis útikalauz stopposoknak alkotója szerint ez egy kis sárga hal, amit ha a fülünkbe teszünk, bármilyen nyelvet azonnal és folyamatosan fordít



3. kép Wordtrans – személyes fordítószótárunk



4. kép Az Editct felülete

arra a nyelvre, amelyiken beszélünk. De ahogy Adams is írja: „Mindez nem akadályozta meg, hogy a szegény Bábel-hal hatásosan lerombolja a különféle fajok és kultúrák közti összes kommunikációs akadályt, és így több és véresebb háború okozójává váljék, mint bárki vagy bármi más a történelem folyamán.” (Molnár István fordítása)
A KDE webböngésző, fájlkezelő és svájcibicska keresztezésével született programja, a Konqueror beépített horoggal rendelkezik az AltaVista Bábel-halához. Csak látogassunk el egy idegen nyelvű oldalra – könnyedén lefordíthatjuk az ott talált információkat. Én például véletlenszerűen meg-

KAPCSOLÓDÓ CÍMEK

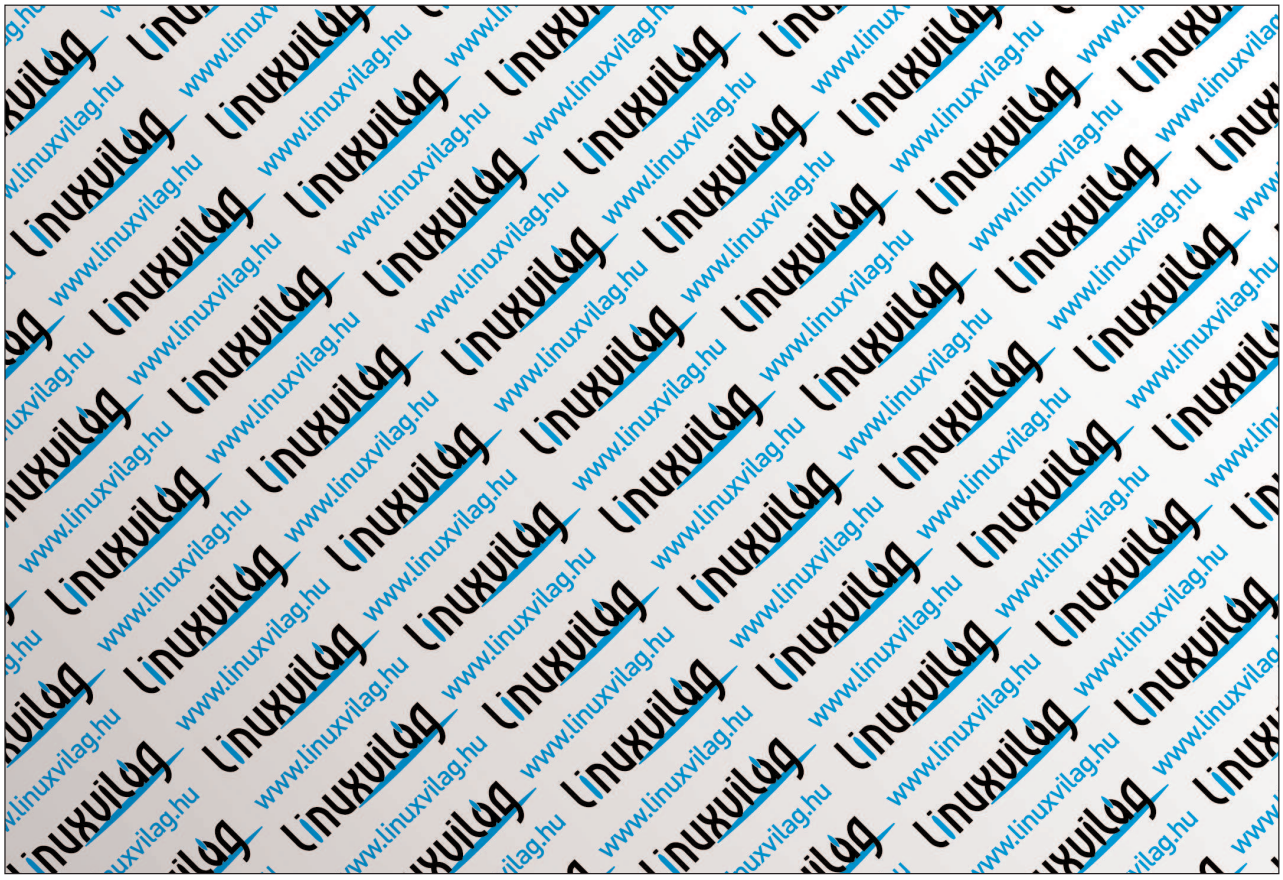
Edict ➲ http://edictionary.sourceforge.net
A KDE honlapja ➲ http://www.kde.org
A Kdict honlapja
➲ http://www-user.rhrk.uni-kl.de/~gebauerc/kdict
Az MWSpeak honlapja
➲ http://www.jclement.ca/Projects/mwspeaker
TuxFinder ➲ http://www.tuxfinder.org
Wordtrans ➲ http://wordtrans.sourceforge.net
Marcel borlapja
➲ http://www.marcelgagne.com/wine.html

látogattam a Die Welt német nyelvű magazint, amiről megtudtam, hogy azt jelenti: „A világ”.
Amikor az oldalunk betöltődött, kattintsunk a Konqueror ***Tools*** (eszközök) menüpontjára, válasszuk a ***Translate Web Page*** (weboldal fordítása) lehetőséget, és válasszunk nyelvet a legördülő listából. Én a „German to English” lehetőséget választottam, et voilà! Máris olyan nyelven olvashatjuk a szöveget, aminek számunkra jóval több értelme van.
A saját személyes és helyi fordítószótár érdekében okos dolognak tűnhet egy pillantást vetni ***Ricardo Villalba*** Wordtrans nevű programjára a ➲ http://wordtrans.sourceforge.net címen (lásd a *3. képet*). A program lefordítása egy kicsit trükkös is lehet, de nagy gondot nem okozhat. Ellátogatva a TuxFinder honlapjára (➲ http://www.tuxfinder.org) jócskán találhatunk előre lefordított csomagokat is. Letöltöttem a Wordtrans alapcsomagját, a Wordtrans-kde RPM-csomagot, és telepítettem is őket. Ha csomagokat is le szeretnénk tölteni, akkor mindkettőre szükségünk lesz. Találkozhatunk a Wordtrans-qt és Wordtrans-web csomagokkal is. Alapesetben a Wordtrans angol, francia, olasz, portugál és spanyol szótárakkal érkezik, de továbbiak is hozzáadhatók.
A Wordtrans honlapján és a programban is találhatunk egyéb nyelvi fájlokra mutató hivatkozásokat. Amikor elindítjuk a Kwordtrans programot, úgy tűnhet, hogy semmi sem történt, de pillantsunk csak a Kicker panel rendszertálcájára – itt észrevehetünk egy kis szürke könyvet ábrázoló ikont. Erre rákattintva megjelenik a Kwordtrans felhasználói felülete. A kívánt nyelv kiválasztásához kattintsunk a menüsor ***Dictionaries*** (szótárak) pontjára, és válasszunk egyet a listából. Válasszuk ki a fordítás irányát (például angolról spanyolra, spanyolról angolra), írjunk be egy szót, és nyomjunk ENTER-t.

Mint már említettem, további szótárak hozzáadására is mód van, ehhez a menüsor ***View*** (nézet) menüjének ***Introduction*** (bevezetés) pontját kell a lehetséges hivatkozások végett kiválasztanunk. Ez nem csak a Wordtrans képességeinek a kiterjesztését jelentheti, léteznek olyan letölthető szótárak is, amik alaposabbak, mint az alapértelmezett fájlok. Egy letöltött nyelvi fájl hozzáadásához a menüsor ***Dictionaries*** (szótárak) ***New*** (új) pontját kell kiválasztanunk, és követnünk kell a megjelenő utasításokat. Számos szótárat sikerrel töltöttem le a ➲ http://www.linuks.mine.nu/dictionary címről, és kifogástalan eredménnyel telepítenem is sikerült őket.
Attól tartok, hogy nem beszélek magyarul, mes amis, de szemmel láthatóan azt kell mondanom, hogy „az idő lejárt”, ami az asztali tolmácsom szerint annyit jelent, hogy az idő eljárt. Csakugyan itt a záróra, de még egy pohár borra azért van időnk, mielőtt elmennétek. Tartsátok a poharaitokat, és hűséges pincérem, François, boldogan kiszolgál benneteket. Látjátok, egy kis felfedezőút Linux-konyháinkban, és egy nap talán minden erőfeszítés nélkül társaloghatunk a világgal.
A következő alkalomig, mes amis, ürítsük poharainkat egymás egészségére. A votre santé! Bon appétit!

Linux Journal 2003. július, 111. szám

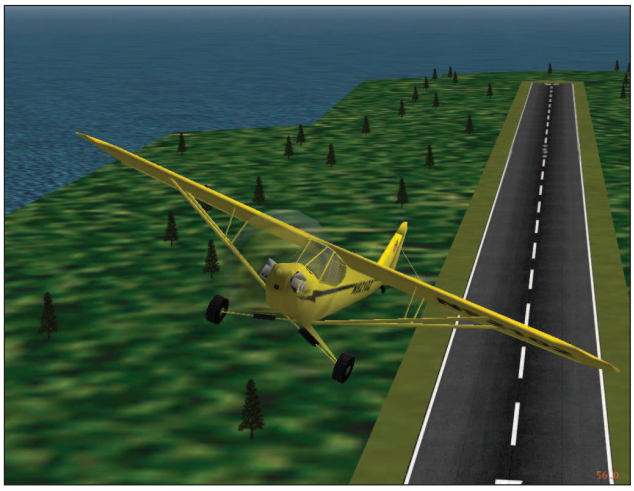
Marcel Gagné (mggagne@salmar.com)
Mississaguában, Ontario államban él.
Ő a szerzője a Kiskapu kiadásában tavaly szeptemberben megjelent Linux-rendszerfelügyelet (ISBN 96-9301-40) című könyvnek (jelenleg is egy könyvön dolgozik).



Játékmustra

A fejlesztők mostanában sem tétlenkedtek, és újabb játékokkal örvendeztették meg a Linux-felhasználókat. Ezek közül szemezgettünk.

A FlightGear egy körülbelül négy éve indult projekt, amelynek a célja nem kevesebb, mint egy nyílt forrású polgári repülőgép-szimulátor létrehozása, ami különböző felületeken képes futni. Noha az első üzembiztos változat még mindig nem jelent meg (jelenleg a 0.9.2-es változatnál tart), nem kell már sok hozzá, hogy felvehesse a versenyt kereskedelmi forgalomban kapható társaival.
Ebben a játékban minden megtalálható, amit egy ilyen szimulátornak tudnia kell. Több választható géptípussal száguldozhatunk – a Cesnától kezdve a Boeing 747-esen keresztül egészen az űrrepülőig. Többjátékos módra is lehetőség nyílik, ám ez a része még nem túl kiforrott. A grafikára egy rossz szavunk sem lehet, olyan apróságokra is odafigyeltek, hogy az ég színe a napszakoknak megfelelően valósághű legyen. A repülésnél az egyik legfontosabb meghatározó tényező az időjárás, amit a felhasználó a korábbi változatban szerepelthez képest sokkal pontosabban is beállíthat. Véletlenszerű időjárást is választhatunk, ekkor meglepetés lesz, hogy milyen viszonyok között kell repkednünk.



A FlightGear azonban még korántsincs készen. Továbbra is zajlik a hibajavítás (bugokból sajnos még mindig szép számmal akad), illetve az apróságok csiszolása. Ha egyszer teljesen elkészül (ez remélhetőleg a közeljövőben várható), akkor talán az egyik legrészletdúsabb repülőszimulátort tisztelhetjük e játékban. Sőt ez már nem is csupán játék, hiszen akár pilóták kiképzésére és más „komoly” feladatokra is használható.
A nyílt forrás pedig csak használ a FlightGear efféle felhasználásának, hiszen bárki szabadon kiegészítheti, bővítheti a program tudását.
➲ http://filghtgear.org

LBreakout2

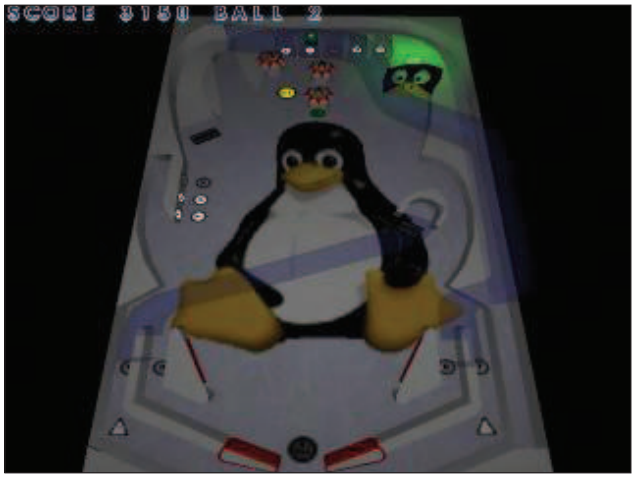
A „break-out” talán a legősibb számítógépes játékok egyike, biztos mindenki játszott már ilyennel. A feladat az, hogy az összes téglát úgy bombázzuk szét egy pingponglabdával, hogy a labda ne essen le, hanem visszapattanjon a képernyő alján általunk irányított lapról.
A break-out játékok népszerűsége nem csökkent, máig is kedvelt időtöltés unalmas géptermi órákon, illetve a főnök távollétében a munkahelyeken. Sokféle képviselője van ennek a stílusnak, mind közül azonban talán az LBreakout2 a legprofibb.
Nemcsak az ötven végigjátszható pálya, a könnyen kezelhető pályaszerkesztő, illetve az Internetről letölthető témák (amelyek egyéni külsőt s hanghatásokat kölcsönöznek) miatt érdemelte ki ezt a címet, hanem azért is, mert rengeteg olyan ötlettel pumpálták tele, ami színesíti, ugyanakkor nehezíti a játékmenetet. Ilyenek például a fentről hulló rendkívül sokféle csomagocskák – ezeket felszedve hol pozitív, hol negatív hatások érnek minket: például pluszpontot kaphatunk, megnőhet az ütő hossza, egy további életet nyerünk, esetleg még egy labdát (ez jó, mert ha az egyik le is esik, még mindig ott van a másik). A legjobb dolog, ami történhet velünk, hogy alul egy fal képződik, amiről a labda visszapattan (a mélybe esés helyett). Így a játékosnak semmi dolga nem akad, csak meg kell várnia, amíg a folyamatos pattogásban lévő labda az összes téglát szét nem veri.
Természetesen kevésbé előnyös hatások is érhetnek minket, ezek közül a legötletesebb az elsötétedés, amikor a téglákat nem látjuk, csak a labdát és az ütőt. Ennek egy kicsit nehezebb változata, amikor az ütő is eltűnik, így csak sejtéseink lehetnek a pontos helyéről.
Téglából is rengetegféle van. Létezik olyan, amit eltalálva az egyszerűen szétesik, de akad olyan is, amelyik hatalmas robbanás kíséretében semmisül meg, a környező téglákat is magával rántva. Olyan tégla is akad, amit nem elég egyszer eltalálni, hanem kétszer vagy akár háromszor le kell bombáznunk.
Az LBreakout2 utolsó szót érdemlő szolgáltatása a többjátékos mód, ami lehetőséget ad arra, hogy a megszállottak Interneten keresztül is összemérhessék az erejüket.
➲ http://lgames.sourceforge.net/index.php?project=LBreakout2

Emilia Pinball

Ha flipperezni támad kedvünk, ám lusták vagyunk lemenni az első kocsmába, akkor az Emilia Pinball a mi játékunk! Igaz, a valódi élményt ez sem pótolja, sőt valószínűleg nem ez a játék a legszórakoztatóbb ebben a műfajban, de legalább fut Linux alatt. Fél éjszakát ezzel is el lehet szórakozni, főként ilyen forróságban kitűnő kikapcsolódás.
➲ http://pinball.sourceforge.net/

TuxType2

Nagyszerű segédprogram, segítségével csemeténk játszva összebarátkozhat a billentyűzettel. A TuxType2 többféle játékmódot tartalmaz, amelyekkel gyakorolhatjuk, hogy melyik gomb hol helyezkedik el a billentyűzeten. Az egyik közülük a „pingvinetetős”: a képernyőn fentről lefelé halak úszkálnak, mindegyikükön egy-egy betű szerepel. Ha lenyomjuk a betűhöz tartozó billentyűt, pingvinünk odasétál a hal alá, és amikor



az leér, menthetetlenül megeszi. A cél, hogy ilyen módon az összes halat megetessük Tuxszal.
Létezik egy másik játékmód is, ebben a halak már szavakkal „érkeznek”, amiket annyi idő alatt kell begépelnünk, amíg le nem érnek a képernyő aljára. Mivel a nehézségi fokozat állítható, még azoknak is kihívást jelenthet, akiknek a keze alatt füstölni szokott a billentyűzet.
➲ http://tuxtype.sf.net/

Xrick

Még a Lara Croft előtti időkben, a nyolcvanas és a kilencvenes évek elején volt egy név, amire minden „mászkálós” játékok iránti rajongó felkapta a fejét. Ez a név pedig a Rick Dangerous volt. Rick közeli rokonságban állt Indiana Jonesszal, csak míg az utóbbi a mozivásznakon menekült lezúduló kövek és gyilkosabbnál gyilkosabb csapdák elől, addig Rick a számítógépes játékokban tette ugyanezt.
Az Xrick is azoknak az időknek a hangulatát idézi mind a grafika, mind a hanghatások terén. Ez úgy értendő, hogy a grafika körülbelül az alap-Nintendo csúcsteljesítményével vetekszik, a hanghatások pedig némi buta pittyegésben kimerülnek. Mindenesetre azok számára, akik nosztalgiázni szeretnének vagy már unják a 3D-s csodákat, kihagyhatatlan darab.
➲ http://www.bigorno.net/xrick/

Majesty

Végre elkészült a Majesty Linux alatt futó változata is.
A játék sajnos nem ingyenes, viszont a játszható bemutatópéldány szabadon letölthető a
➲ http://demos.linuxgamepublishing.com/majesty/ oldalról.
A Majesty nem pusztán egy valós idejű stratégiai játék. A fejlesztők fantasybirodalom-szimulátornak nevezik. Készítői valójában a stratégiai elemeket ötvözték a szerep- és kalandjátékokban megtalálható elemekkel.
Feladatunk az Ardiana nevű királyság irányítása. Ez nem merül ki az építgetésekben, például rendeleteket is kell hoznunk, amelyek erősen befolyásolják birodalmunk lakóinak a viselkedéséi. A lakosságot nemcsak rendeletekkel, hanem épületek felhúzásával is befolyásolhatjuk. Ha például sok kocsmát vagy játéktermet építünk, javul a közhangulat, és így kevésbé lesznek érzékenyek az adóemelésre.
De ez még mind semmi! Segítségünkre vannak az úgynevezett hősök is, akiknek feladatokat adhatunk. Minden hős más-más személyiségjegyekkel bír, és gyakran önállóan is cselekszik. Ha hősünk személyisége olyan, akkor könnyen elkalandozhat, és semmibe veheti a kitűzött feladatot.
A hősök viselkedését azonban rendeletekkel vagy épületek felhúzásával szabályozhatjuk. Ha például iskolákat, könyvtárakat telepítünk, akkor a hősök megfontoltabbak lesznek, így küldetéseiket jobban végre tudják hajtani.
A Majesty tehát egy összetett, ugyanakkor szórakoztató és érdekes játék. Minderre rátesz a gyönyörű szép grafika és a hanghatások, amelyek sajátos hangulatot adnak a játéknak.

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

Játszótér

E havi témánk a múlt hónapból öröklődik:
a Quake grafikai motorjára épülő játékok leírását folytatom.

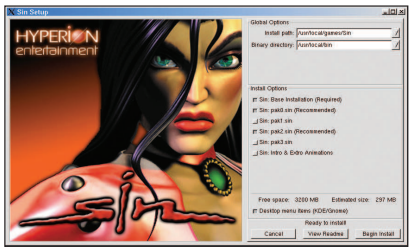
Istenem, „bűn” lenne kihagyni a SiN-t. A SiN játék – mint neve is utal rá – a bűnről, valamint annak üldözéséről szól.

A SiN története

Tíz évvel ezelőtt, 2027-ben Freeport az összeomlás szélén állt. Az eredménytelen rendőri erők elvesztették a hosszú, kemény harcot a bűn ellen. A városnak segítségre volt szüksége, a túlterhelt kormányzat nem tudta fenntartani a békét az utcákon. A kormányzat új törvényjavaslatot indítványozott, ami egy önálló, a saját biztonsági erejüket képező testület felállítását engedélyezte, ennek egységei a másoderők. Ezek a másoderők alkalmasak rá, hogy gondoskodjanak a városi tanács és a város polgárainak személyes védelméről.
A HARDCORPS segítségével a káosz lassan alábbhagyott a városban, visszatért a béke és a rend. John Blade ezredes, a HARDCORPS tulajdonosa és csapata pillanatnyilag a veszélyes U4 drog forrását vizsgálja. Ebben JC a segítőtársa.
Jelenleg a drog több helyen előfordul – használói nemrég veszélyes dolgokat tapasztaltak vele kapcsolatban. Blade megpróbálja kideríteni, ki és mi áll e megmagyarázhatatlan jelenség mögött, és megpróbálja megállítani a terjedését. A bonyolult kirakósdi lassan összeáll, és egy pont felé mutat, a briliáns biokémikus, Elexis Sinclaire felé.
Elexis talány a káprázatos külső mögött; olyan ellenzéki képviselő, aki megmutatja, hogy az üzleti világban nincs kegyelem. A háttérben dolgozik, rejtélyes boszorkánykísérletet végez az emberiség evolúciójának és ezeréves történelmének a megállítása végett, de semmi sem sikerül neki, amit eltervez, mert Blade megakadályozza benne.

Telepítés

A telepítőprogram használatakor megadható, hogy milyen mennyiségű állományt szeretnénk felrakni a gépünkre. Tapasztalatból kell mondanom, hogy a teljes telepítés teljesen fölösleges, mivel a program CD nélkül nem fog futni, így



ha valakinek elég gyors a CD-meghajtója, akkor elégséges csak az ajánlott tételeket felrakni a gépére. Furcsa, hogy míg az eredeti windowsos változat nem rendelkezik szériaszámmal, addig a linuxos változatot a Hyperion szériaszáma védi.
A telepítést követően a program elsőre nem fog futni, mert régi 1.1 változatú SDL-fájlokat keres. Nincs semmi gond, csak hivatkozást kell készítenünk a meg-

lévő SDL fájlra, méghozzá a program által keresett néven. (A pontos fájladatra akkor jöhetünk rá, ha a programot `xterm` alól a SiN saját linuxos könyvtárában kiadott paranccsal próbáljuk meg futtatni.) A másik furcsaság az, hogy noha a program változatszáma 1.10-es, ennek ellenére nem programozták le benne a Wages of SiN küldetéslemez támogatását. A küldetéslemezt egyszerű másolással fel lehet rakni (2015 könyvtár, valamint a ***pack4.pak*** fájl), de mi haszna, ha sem a zseblámpa, sem az infraszemüveg, sem a pluszfegyverek nem használhatók, mert meg sem jelennek? Ugyanúgy, akárcsak a kötelek és más tárgyak, amik jellemzően csak a Wages of SiN-ben találhatók meg. Pedig mindehhez csak az 1.03-as változat kellene – sajnos ezekről az élményekről így lemaradunk.
Bevallom, mindezek ellenére ez a program az egyik kedvencem.
A történet – a Soldier of Fortune-nel ellentétben – többszálú. Ebben ugyanis, ha valamit nem gyűjtünk be vagy nem sikerül megszerezni, megölni, akkor másként folytatódik a történet. Természetesen azért a 4–5. alkalomra elég kiismerhetővé válik a program, de még ezután sem lehet megunni. Vajon miért kellett kihagyni a Wages of SiN-támogatást? Ki tudja?
A felhasználható fegyverek listája:

- Magnum
- vadászfegyver
- orvlövész puska
- impulzuságyú
- három karomlábú akna.

Soldier of Fortune

A program ***John F. Mullins*** karakterére épül. A játék történetét részben John életének pillanatai szövik át. Főhősünk járt Vietnamban, részt vett egy pár diverzáns akcióban, 21 év szolgálat után fejezte be katonai pályafutását és közben szépen feldíszítette az egyenruháját. Tanácsadóként részt vett a program kialakításában is.
A játékban a következő fő karakterekkel találkozhatunk:

- ***John Mullins*** Délnyugat-Oklahomában született, három bevetésen vett részt Vietnamban, és háromszor kapta meg a „Bíborszív”-et. Tagja volt a vitatott ***Phonix*** csoportnak, tagjai magas beosztású vietkongok ellen követtek el merényleteket. Mullins a hadseregből történő visszavonulását követően konzultánssá,

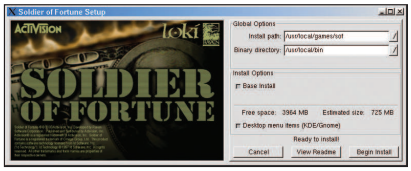
még inkább zsoldossá vált a 12 év alatt. Jelenleg a U.N. antiterrorista csoportnak dolgozik, amit csak „A Bolt”-nak neveznek. Normál felépítése ellenére Mullins büszke az erejére és gyorsaságára, ami több férfinak is dicsőségére válna.
- ***Aaron „Hawk” Parsons*** Detroit közepén nőtt fel, később harcolt a Sivatagi Viharban. Parsons a hadsereg egyik megfontolt robbantási szakértője volt. Gyakori partnere volt John Mullinsnak „A Bolt”-ból érkező munkák esetén.
- ***Sam Gladstone***, könyvesbolt-tulajdonos, részmunkaidős alkalmazottja „A Bolt”-nak. Kapitányként megjárta Vietnamot, remek felderítő volt, később néhány évet a CIA-nál töltött. 62 éves és szakállas.

Milyen fegyverek állnak a rendelkezésünkre?
- harci kés (henteskés). Fontos tulajdonsága, hogy nemcsak kaszabolni lehet vele, hanem eldobva távoli ellenségeket is ártalmatlanná lehet tenni, hang nélkül.
- 9 mm-es pisztoly (fekete párduc)
- nagy pisztoly (ezüst karom). Ezüst színű és sokkal nagyobb hangú és pusztító erejű, mint a „kistestvére”.
- vadászpuska (vad harcos)
- orvlövész puska (sasszem)
- gépkarabély (raptor)
- hangtompítós SMG (bulldog)
- nehéz géppuska (M-75 őr)
- M202A2 „fényjelző” rakétavető
- M343D lángszóró
- H-24 henger alakú puskagolyóvető
- mikrohullámú impulzusfegyver (MP).

A történet sajnos nem mondható igazán túlkombináltnak, egy-egy pályát addig úgysem lehet elhagyni, amíg a feladatot nem teljesítettük. Furcsa, hogy mindenért pénzt kapunk, amit semmire nem költhetünk el. Ennek ellenére szórakoztató és a Quake 2 grafikai motorjából az összes különlegességet kihozó játékkal ülünk szemben. Ez a Ghoul (vámpír) leképezési rendszer érdeme, ami nem más, mint a Raven-kiegészítése a Quake2-motorhoz. Mit köszönhetünk neki? Az ellenségek néha olyan vérfagyasztóan, hörögve halnak meg, hogy gyengébb idegzetű játékosok talán félve néznek a képernyőre. Előfordult, hogy egy-egy agonizáló ellenfél hörgése betöltött egy egész termet, és messziről is visszhangzott a (például a metróban).

Néhány szó a telepítésről

A telepítő 725 MB-ot tesz fel a gépünkre. Telepítés közben nincs egyéb lehetőség, viszont a játék indulását követően beállítható a „szülői” kontroll, amivel



koordinálni lehet a hörgések és a vér mennyiségét a játékban.
A telepítést követően az első indításkor nekem egyetlen hang sem jött ki a gépből, így felraktam az 1.06a szervizcsomagot (a CD-mellékleten megtalálható), ami azzal együtt, hogy kijavítja az adott hibát, a Gold változatra frissíti a programot. Szintén megtalálható mellékletünkön a ***sofmappack.zip***, amit a ***Base*** könyvtárba kicsomagolva a telepített SoF-ot Platinum változatúvá tudjuk varázsolni. A Platinum változat gyakorlatilag csak új pályákat tartalmaz: airportdm, factorydm, irqdesertdm, labdm, plantdm.
Némi ízelítőt is adunk a Platinum változat nyújtotta pályák képi világából. Következzen az erre a hónapra valójában eltervezett anyag: a nagy vetélytárs a játékok között, az Unreal és annak változatai.

Unreal

1998-ban jelent meg az első változat, ami már kiadásakor megosztotta a játékos nagyközönséget.
Gyakorlatilag két részre szakadt a tábor: akadt, aki ki nem állhatta, és volt, aki istenítette és a fellegekbe emelte a látvány és játékélmény miatt. Bevallom, én kezdetben azok közé a kevesek közé tartoztam, akik ki nem állhatták a programot. Ez akkor változott meg, amikor a fejlesztők elkezdték a javítócsomagokat kiadni a programhoz, és a végleges 2.2.6-os változattal végre egy olyan játékot tudtam futtatni, ami támogatta TNT alapú kártyámat.

A történet szinte felejthető. Igazából nem ez fogja meg a játékost, bár a „mentsük meg a bolygót az idegenektől” jellegű történet önmagában nem jelentene nagy kihívást, sokkal többet nyom a latban a képi világ, amit a programozók megalkottak. Én azért nem szerettem a programot, mert tele van kis mütyür alakokkal, de mihelyt nemcsak a 3Dfx tulajdonosok tudtak a programmal 3D-ben játszani, máris nyilvánvalóvá vált, hogy a cselekményhez adott körítés, a pályák és a grafika együttese mindenkit el tud varázsolni. Fedezzük fel ezt az irreális világot!

Telepítés

A ➲ http://www.icculus.org oldalról letölthető a telepítőprogram (***unreali-install.run***), ehhez csak a windowsos CD szükséges. CD-mellékletünkre felkerült a szükséges állomány.

Szükséges elemek

1. A program futtatásához sajnos szükség van egy Unreal Tournament játékra, aminek telepítve kell lennie a Linuxon.
2. Mielőtt elindítjuk a telepítőprogramot, az Unreal CD tartalmát be kell fűzni.

3. Ha a telepítő nem találja a telepített UT-t, akkor a következő jellemzőkkel meg kell adni neki:
```
export UT_INSTALL=/valós/
telepített/ut
```
4. Hogyha rendelkezel egy windowsos lemezrésszel, és az ott telepített Unrealt szeretnéd futtatni Linux alatt, akkor:
```
export SETUP_CDROM=/mnt/
➥windows/Program\
➥Files\Unreal
```

Mivel a program telepítéséhez kell az UT, valamint a telepítőprogram is egy módosított OpenUT-t teszt fel a program indításakor, nem az Unrealra jellemző képernyő és menüszerkezet fogad minket, hanem az UT-é.

Unreal Tournament

1999-ben megjelent a folytatás, ami az Unreal nagyon is hiányos hálózati és a többjátékos módot érintő hiányosságait igyekszik pótolni, illetve befoltozni.
A program nem kapott új grafikai motort, csak a régit porolták le a fejlesztők, egy kicsit belenyúlva ebbe-abba, és máris új néven került polcra a termék. Furcsa ez azért is, mert valójában olyan jellegű történettel, mint az Unreal, nem találkozhatunk. Miként a Tournament szó is mutatja, egy bajnokságról, versenyről van szó. Egyetlen teendőnk, hogy a versenyeket – egyedül vagy egy csapat részeként – meg kell nyernünk. Kezdetben sokan megkérdőjelezték, hogy van-e egyáltalán létjogosultsága egy ilyen programnak, főként kis hazánkban, ahol otthon nincs hálózat, és a program kiadásakor az Internet lassú és drága volt (igaz, most sem sokkal rózsásabb a helyzet).
Fontos megjegyezni, hogy a linuxos változat nem létezik dobozos kereskedelmi termékként. A Loki legalábbis nem adott ki ilyet, ezért a ➲ http://www.tuxgames.com-on megvásárolt CD-n a windowsos, teljes kereskedelmi változat, valamint a Loki által készített telepítő található.

Telepítés

Le kell tölteni az ***ftp://ftp.lokigames.com/pub/patches/ut/ut-install-436.run*** fájlt, és rendszergazdai jogosultsággal futtatnunk kell. A telepítőprogram keresi a windowsos CD-t, így annak benne kell lennie a meghajtóban, és be is kell lennie fűzve (természetesen a szükséges állomány CD-mellékletünkön szintén megtalálható). Mellékletünkön szerepel egy biztonsági frissítés is, ami a denial-of-service hibát javítja ki (***IpDrv-436-Linux-08-20-02.zip***).
Teendőink az alábbiak lesznek:

- Le kell cserélni a gépen lévő ***IpDrv.so***-t a zipfájlban lévővel (biztonsági mentés vagy tartalékmásolatkészítést követően).
- Belátásunk szerint a már meglévő ***/home/loki/ut/UnrealTournament.ini*** fájlhoz hozzá kell adni az alábbiakat (a ***IpDrv.TcpNetDriver*** részben):
```
AllowPlayerPortUnreach=
➥(True or False)
➥LogPortUnreach=(
➥(True or False)
```

Az alapértelmezésbeli beállítások elsőre megfelelnek. A program tökéletes futásához feltétlenül ajánlom a legalább 128 MB RAM-ot, és érdemes odafigyelni a tárhelyre is, mert csak a telepítés 550 MB-nyi helyt foglal el, és még el sem kezdődött a játék, sem a pályamentések! A telepítést követően nem kell a korong, a program nélküle is tökéletesen fut a megfelelő erősségű gépen. Márpedig a program gépigénye – sajnos – nagy, ezzel tisztában kell lennie annak, aki az UT futtatására vállalkozik. Lassabb kártyákon és gyengébb gépeken segít, ha 16 bites színmélységet használunk, mert így felgyorsul a játék sebessége.

Ezt viszont sajnos nem lehet a programból állítani, mint a Quake3-ban, hanem a grafikus felületet kell beállítani.
A legelső indításkor előfordulhat, hogy a játék szaggat és lassú, szintén a gyengébb gépeken; ilyenkor a ***/home*** könyvtárban létrejön egy ***/loki/ut/UnrealTournament.ini*** állomány, amit érdemes végigböngészni, és a megfelelő helyen („....=libGL.so”) a VGA kártya OpenGL meghajtójára mutató hivatkozásra lecserélni, például ***/usr/lib/libGL.so***.
A Gnome és KDE esetében is beköltözik egy ikon a menübe, amit ki lehet húzni a munkaasztalra.
A játékhoz feltétlenül tudom ajánlani egy USB-s egér beszerzését, különben a jellemzőknél nagyon fel kel húzni az egér gyorsítását. Egy dolog azonban biztos: az

UT grafikailag tökéletes, és nem véletlen, hogy annyian (többek között én is) többre tartják a program grafikai megjelenítését, mint a Quake3-ét!

UT2003

2002-ben kiadták az Unreal Tournament 2003-at. Már valódi linuxos anyagról beszélhetünk, olyannyira, hogy a 3 CD-ből álló telepítőkészlet utolsó lemezén található meg a telepítőprogram. Ügyesen oldották meg, mivel a program azzal

kezdi a folyamatot, hogy átmásolja magát a linuxos rendszerre, majd onnan fut tovább. Ez azért is szükséges, mert a linuxos telepítéshez is kell mind a három korong – ezeket a telepítőprogram szépen sorrendben be is fogja kérni.
A telepítés megkezdésével együtt vegyünk magunkhoz egy nagy kanna kávét, bátrabbak valami erősebbet is előkereshetnek, mivel a telepítési folyamat idegőrlően sokáig tart.
Az UT egyedi megoldásokat próbált felmutatni, a 2003-as változat különböző játékokból kísérel meg meríteni, ám mégsem érezzük azt, hogy megváltoztatná a régi élményvilágot. A grafika magáért beszél, azt hiszem, szükségtelen jellemeznem a minőséget, illetve látványvilágot, mert a játékosoknak biztosan van fogalmuk róla, hogy milyen csodálatosan néz ki az Unreal Tournament 2003. Az új játékmotor, a nagy

Játékhírek

Enemy Territory

Megjelent a Return to Castle Wolfenstein folytatása, illetve küldetése, az Enemy Territory. Mivel ez még nem az „igazi" végleges kiadás, csak a nagy sávszélességgel rendelkezők töltsék le. Az egyjátékos üzemmód teljesen kimaradt belőle. Mivel az ➲ ftp.idsoftware.com tárhelyről hiányzik az anyag, a tartalmáért nem az idSoftware a felelős.
➲ http://www.castlewolfenstein.com/

America's Army 1.7.0

Megjelent az America's Army 1.7.0 kiadása. A játék egy taktikai program, amit az amerikai hadsereg is támogat. Újabban divat, hogy teljes játékokat adnak közre, a játékosok legnagyobb örömére. A teljesen ingyenes program letöltéshez sok idő és nagy sávszélesség szükséges, mivel a teljes anyag 316 MB.

Half-Life

A Linuxos Half-Life-ügyfél petícióhoz a ➲ http://www.tuxgames.com/petitions/ oldalon lehet csatlakozni. Évekkel az eredeti játék kiadását követően most került sor arra, hogy valakinek végre eszébe jusson, hogy ezt a sokak által közkedvelt programot át kellene ültetni Linuxra. Jelen pillanatban

csak kiszolgáló működik Linux alatt. Egyéb futtatási lehetőség csak Wine vagy WineX használatával lehetséges.

WineX 3.1

Örömmel jelenthetem be, hogy a Transgaming hozzáférést engedélyezett az anyagaihoz. Így a WineX-ről egy minden részletre kiterjedő anyag fog megjelenni a következő számban. A rossz hír az, hogy a Transgaming semmilyen formában és semmilyen lehetőséget nem nyújtva a megoldásra megtagadta, hogy a programot a CD-mellékleten közöljük.

Neverwinter Nights

Hosszú várakozás után végre megjelent a Neverwinter Nights végleges ügyfélváltozata. Az ügyfél még csak *tar.gz* formátumban létezik, így a telepítéshez szükség van egy windowsos NWN CD-re, valamint egy érvényes CD-kulcsra is. A Tuxgames ígéretei szerint egy-két héten belül megjelenik a felhasználóbarát változat, ami feltehetőleg

Loki-alapokon nyugvó telepítőt, valamint teljes CD-anyagot fog jelenteni. A program a kiadást követően egyből változatszámot is váltott 1.30-ra. A folt 40 MB méretű, és elsősorban a videolejátszási gondokat orvosolja (a videofájlok nem éppen Linux-megfelelők). Egyúttal bejelentették, hogy az NWN küldetéslemezt a Shadows of Undertide is támogatja a linuxos ügyfélhez.
➲ http://nwn.bioware.com/downloads/linuxclient.html

felbontású textúrák gondoskodnak róla, hogy minden élethűbbnek látszódjék. Ezért viszont fizetnünk kell, és nemcsak a pénztárnál, de a PC-s szaküzletekben is. Étvágya ugyanis egy 1 GHz-nél nagyobb processzort és egy GeForce3-erősségű kártyát igényel, ennél régebbi és gyengébb eszközön a program majdhogynem játszhatatlan.
A játékmódok egy újjal bővültek, a „bombing run"-nal – lásd az amerikai futballt. Körülbelül ehhez hasonló a felépítése, ugyanis a labdát meg kell szerezni, és az ellenfél kapujába kell juttatni. Lehet vetődni, rúgni stb. pontosan olyan módon, mint az amerikai fociban. Apróbb módosítás, hogy a Domination esetében 10 mp-ig nálunk kell lennie a zászlónak, különben a csapat nem kap pontot.
Megszokott fegyvereinkkel a folytatásban is találkozhatunk, igaz, a modellek elég nagy fejlesztésen mentek keresztül. Elsősorban grafikai módosításokról beszélhetünk, mivel maga grafikus alrendszer is megváltozott. Természetesen akadnak használatbeli módosulások is, például a „translocator"-t (teleport) egy kamerával látták el, ezzel követhetjük, hogy mi történik a teleport során. Új fegyver a villámvető (lightening gun), ez egy hatalmas villámnyalábot lövell ki magából, ezzel bénítva meg a gaz ellent. Másodlagos fegyverként egy távcsöves (sniper) puskát kapunk. Létezik egy újabb fegyver: a Shield Gun, nevéből fakadóan inkább pajzsnak felel meg.
Az UT 2003 is tartalmaz gépi ellenfeleket (bot), amikkel helyi hálózat vagy Internet nélkül tehetjük próbára a tudásunkat. Ezeknek a botoknak az intelligenciája nagy fejlődésen ment keresztül. Igaz, ezzel már az UT esetén sem volt gond. Gyakorolni pedig a több mint harminc pályán van hely, amikből néhány ugyan ismerős lehet az UT-ből, de még ezeknek is átalakították grafikai megjelenítését. A látottak a legnagyobb részletességgel vannak kidolgozva, mindenféle hatásokkal ellátva (effects). Ha már unalmas lesz a gyári pálya és minden küldetés, illetve a verseny egyre jobban megy, akkor az UT-nak megfelelően itt is rendelkezésünkre áll a pályaszerkesztő és a héjrendszer.
Huh… ezek után mit is mondhatnék? Aki megteheti, mert a munkája megengedi, annak jó nyaralást, aki pedig kénytelen az irodában aszalódni, az vegyen elő egyet a fent felsorolt játékokból, és indulhat a móka (az ebből eredő munkahelyvesztésért a felelősséget azonban továbbra sem vállaljuk).

Kosztadinovszki Norbert
(kosztadinovszki@dialec.hu)
Linux- és játékmániás számítógépőrült.

Ötletbörze

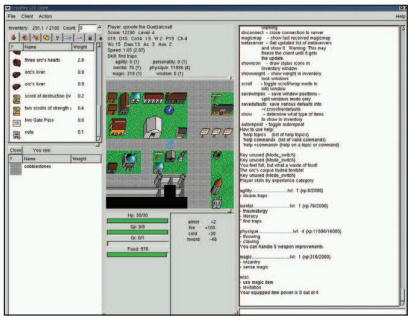
trickle

Gyakran akadhat olyan kevésbé fontos feladat, ami hosszú ideig fut a háttérben (például egy nagy FTP-letöltés). A trickle segítségével a sávszélességet programonként állíthatjuk be, vagyis az SSH-munkamenetek válaszideje gyors marad, mialatt az FTP-letöltés folyik (bár a sebessége kisebb lesz), és a családtagok, illetve a munkatársak sem lesznek idegesek. Az egyetlen hátrány, hogy nem szabad elfelejtenünk elindítani azt a programot a trickle programon keresztül, amelynek a hálózati forgalmát szabályozni akarjuk. Futtatásához libevent, libnsl, libdl és glibc szükséges.

➲ http://monkey.org/~marius/trickle

Crossfire

Ha szereted a szerepjátékokat, ne keress tovább. A Crossfire egy olyan többszereplős, varázslatos szerepjáték, ami hetekre vagy még tovább megakadályozza, hogy értelmes munkát végezz.

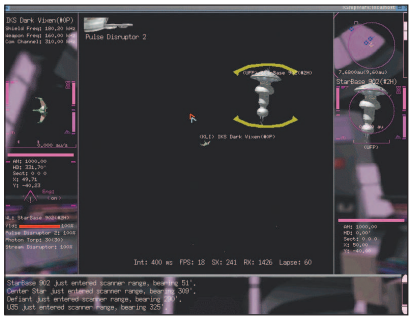


Játszhatsz a világszerte megtalálható kiszolgálókhoz kapcsolódva, vagy saját kiszolgálót is indíthatsz. Egyértelműen sokkal izgalmasabb egy kis csapattal közösen játszani, mint egyedül. Csak ne felejts el enni és aludni néha! Futtatásához X-ügyfél esetén libpng12, libz, libm, libX11, libXext, glibc és libdl; Gtk-ügyfél esetén libpng, libz, libm, libgtk, libgdk, libgmodule, libglib, libdl, libXi, libXext, libX11, libSDL, libpthread és glibc szükséges.

➲ http://crossfire.real-time.com

XShipWars

Három évvel ezelőtt bemutattam a Downloader for X, az Xpuyopuyo és hasonló programokat, de nekem már



akkor is az XShipWars tetszett a legjobban, ami azóta még jobb lett. Igen, ez egy játék, de a családom éppen a játékok miatt nem piszkál azért, hogy másik operációs rendszert telepítsek fel. Az XShipWars (szerény véleményem szerint) alighanem fajtája legjobbja, és folyamatosan fejlődik – nemcsak az ügyfél-, a kiszolgáló- és az olyan kiegészítő programok, amelyek például az univerzumok létrehozásához szükségesek, hanem a rendelkezésre álló univerzumok is, amelyekben játszani lehet.

➲ wolfpack.twu.net/Shipwars/XShipWars

Hasznos ötletek

Szükséged lenne régi weboldalaid pontos másolatára, de most már nem rendelkezel érvényes azonosítóval azon a kiszolgálógépen?
Használd a `wget` parancsot a `--mirror` lehetőséggel.

A */usr* könyvtárat használhatod csak olvasható módban, kivéve, ha programot telepítesz vagy bővítesz. A csak írható, illetve olvasható befűzéseket az `fsck` a befűzések számolása során figyelembe veszi, és az átlagos gépindítási idő csökken. Ha a számítógéped összeomlik, és újra kell indítanod, a */usr*-t az `fsck` nem vizsgálja át.

Nem tudsz belépni SSH-val a gépre? Vagy talán az SSH olyankor kér jelszót, amikor azt hitted, nem fog? Az `ssh -v` parancs használatával bőbeszédű hibaüzenetet kaphatunk az SSH működéséről.

Ha a bash parancsértelmezőben kiadott valamelyik korábbi parancsodat szeretnéd megtalálni, nyomd le a Ctrl-R billentyűkombinációt, valamint néhányat a keresett betűk közül.

Hogyan fordulhat elő, hogy nem működik az IP-álcázás? Mielőtt belefognál a rendszermag újrafordításába, győződj meg róla, hogy az ip-forwarding (IP-továbbítás) valóban be van-e kapcsolva: `cat /proc/sys/net/ipv4/ip_forward`
Ha ez az érték nulla, akkor a következő parancsot használd: `echo 1 > /proc/sys/net/ipv4/ip_forward`

Ha a parancsfájlod összezavarodik, és túl sok HTTP-kapcsolatot létesít a rendszereddel, egyszerűen írd be:
`route add -host hostname`
Ezáltal elkerülheted, hogy a hibás keret teljesen megállítsa a saját rendszeredet.

Ha egy *-rf* elnevezésű fájlt szeretnél törölni, a következőt írd be:
`rm -- -rf`
A -- azt mutatja meg az rm-nek, hogy minden, ami a -- mögött helyezkedik el, az a fájlnévhez tartozik, nem pedig egy kapcsoló.

A Macintosh-textfájlokban található Ctrl-Ms-t a következőképpen tudod átalakítani:
`tr ´\r´ ´\n´ < eredeti > fixed`

Átjátszóvá vált a levélkiszolgálód? Használd az `rlytest`-et annak ellenőrzésére, hogy esetleg mégsem lett az.

Tegyél megjegyzésbe mindent, amit a */etc/inetd.conf* fájlban nem használsz vagy nem értesz.

Egy második DNS-kiszolgálóra van szükséged? Szokás, hogy a Linux-felhasználói csoportok tagjai másodlagos DNS-t szolgáltatnak egymás számára. Érdeklődj egy Linux-felhasználói találkozón!

Linux Journal 2003. július, 111. szám

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