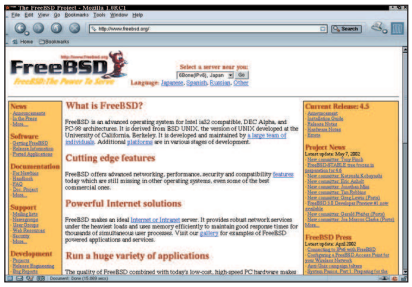
2002 a szabad rendszerek éve

A címben szándékosan használtam többes számot, hiszen nemcsak a Linux érhető el „ingyenesen”, hanem számos egyéb rendszer is.

Biztosan mindenki hallott már valamelyik BSD-alapú rendszerről, melyek közül a FreeBSD a legismertebb. Aki multimédiával foglalkozik, vélhetőleg találkozott már a BeOS rendszerrel, mely szédítő sebességéről ismerszik meg, legyen szó bármilyen multimédiás alkalmazásról. A BeOS sajnos élet és halál között lebeg(ett), azonban lelkes fejlesztők, olyanok, akik a Linuxot a jelenlegi szintre emelték, próbálják menteni a menthetőt, és támogatni kedvenc operációs rendszerüket. Egyéb nyílt forrású fejlesztések is napvilágot láttak, ezek egyike az OpenBeOS. Említésre méltó rendszer a QNX is, ami szintén ingyenesen tölthető le, azonban mind a BeOS, mind a QNX zárt forráskódot használ. Amennyiben valaki arra adja fejét és megvizsgálja a szabad rendszereket, láthatja, hogy a fejlesztők előszeretettel használják a Microsoft által elavultnak ítélt Unix-alapokat (természetesen itt nem említjük meg a Microsoft-termékek kiváltására készülő rendszereket), erre a legfrissebb példa az Apple által megalkotott MacOS X, ami BSD-alapokon nyugszik. A ➲ http://www.freeos.com oldalon körülnézve láthatjuk, hogy nem a Linux az egyetlen választási lehetőség, azonban a támogatottság nagyon sokat számít egy rendszer kiválasztásánál.
A nyílt forrású rendszerek között vitathatatlan tény, hogy a Linux rendelkezik a legjobb a támogatottsággal, ez abból is jól látszik, hogy a BSD-k egy kiegészítés révén linuxos binárisokat is képesek futtatni. A programok jelentős része még mindig GNU-fejlesztés, ezért helyesen GNU/Linux lenne a neve, azonban megjelentek olyan egyéb ingyenes fejlesztések, illetve kereskedelmi termékek amelyek már nem GNU-fejlesztések.
A támogatottságot és az elérhető programok mennyiségét és minőségét tekintve, azt hiszem, elérkeztünk az asztali Linux felnőtté válásához. Napjainkban a Linux telepítése sem kíván meg nagyobb szakértelmet, mint egy akármilyen „ablakos” rendszer, sőt, egy hibásan felismert alkatrész ügyében a linuxot könnyebben rá tudjuk venni a helyes meghajtó használatára, így bizton állíthatom, hogy a kezdő felhasználók sem részesülnek semmilyen megkülönbözetésben, ha szabad rendszereket szeretnének használni. Az általam használt legkényelmesebb grafikus telepítővel a Mandrake Linux és a Red Hat Linux rendelkezik (ezeket használtam, de az is előfordulhat, hogy létezik egyszerűbb is, jobb is). Egy számítástechnikai analfabéta, aki egy kicsit is tud angolul – ez a legtöbb esetben sajnos még mindig alapfeltétele a telepítésnek –, bátran nekiállhat telepíteni az általa kiválasztott Linux-rendszert. A felnőttéválás legszembetűnőbb bizonyítéka a programok 1.0-s változatainak megjelenése. A kedves olvasók nyomon kísérhették az OpenOffice.org, az AbiWord és a Mozilla fejlődését, korongjainkon mindig közzétettük a legfrissebb fejlesztések eredményét, így olvasóink közül valaki önkéntelenül is hozzájárulhatott a fejlődéshez. Így teszünk most is, annak reményében ezeket a csodálatos programokat, hogy egyre több embernek van rájuk szüksége. A Linux éve kis országunk határain belül is érvényes, az FSF csapat hatalmas feladatokat tűzött ki maga elé, amely szintén a kezdő és az angolul nem tudó embereken hivatott segíteni.



Két, egyenként is tekintélyes méretű, de annál fontosabb programot maratoni magyarítási hétvégék alatt használható nyelvi tudással sikerült felvértezni. Az egyik ilyen program az OpenOffice.org volt, a másik pedig a Mozilla, mindkettő – azt hiszem, ezt nyugodtan elmondhatom – minőségi termék, gyors fejlődésüknek és megbízhatóságuknak köszönhetően sokan már a próbaváltozatokat is használták mindennapi munkájukhoz.
A másik igencsak nagy jelentőségű fejlesztés az UHU-Linux, mely teljes egészében anyanyelvünket alkalmazza, így mindenki a kedvére használhatja, nyelvi korlátok nélkül. Reméljük, hamarosan elkészül. Szintén magyar fejlesztés az MPlayer–MEncoder páros, amely a legjobb médialejátszó-, illetve kódoló-program Linux alá – népszerűsége kimagasló. Erről két számmal ezelőtt indított sorozatunkban bővebben is olvashatnak.
Remélem, hogy ez az irányvonal megmarad, és mind több felhasználó csatlakozik ehhez a közösséghez, ha másért nem is, hát önmaguk lelki üdvéért, mivel olyan terméket használnak, ami nem lopott és nem került több tízezer forintba, nem utolsósorban pedig biztonságos is.

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

Kapcsolódó címek

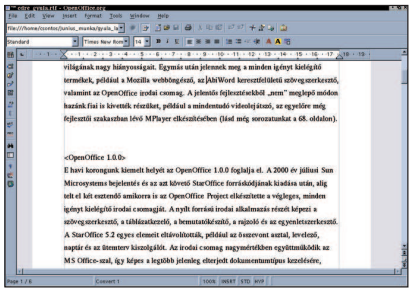
- ➲ http://www.linux.org
- ➲ http://www.freebsd.org
- ➲ http://www.netbsd.org
- ➲ http://www.openbsd.org
- ➲ http://www.qnx.com
- ➲ http://www.beos.com
- ➲ http://www.openbeos.org
- ➲ http://www.atheos.cx

Programvadászat

Talán túlzás nélkül állíthatom, hogy történelmi jelentőségű események szemtanúivá válhatunk. A nyílt forrású fejlesztések eredményeképpen minden igényt kielégítő, a Microsoft monopolhelyzetét megdönteni látszó, de mindenféleképpen jól használható választási lehetőséget kínáló programok jelennek meg a piacon. Miközben a fél világ a jelenlegi antitröszt per útján történő megdöntésére törekszik, addig a nyílt forrású fejlesztések nem is kicsiny hatékony közössége sorra készíti el több rendszeren is jól használható alkalmazásait. A jelenlegi fejlesztések lezártával pótolták a Linux és a Unix világának nagy hiányosságait. Egymás után jelennek meg a minden igényt kielégítő termékek, például a Mozilla webböngésző, az AbiWord keresztfelületű szövegszerkesztő, valamint az OpenOffice.org irodai csomag. A jelentős fejlesztésekből „nem" meglepő módon hazánk fiai is kivették részüket, például a mindentudó videolejátszóéból, és az egyelőre még fejlesztői szakaszban lévő MPlayer elkészítésében is tevékenykedtek (lásd még sorozatunkat a 68. oldalon).

OpenOffice.org 1.0.0

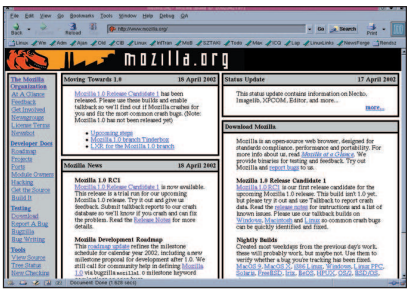
E havi korongunk kiemelt helyét az OpenOffice.org 1.0.0 foglalja el. A 2000 év júliusi Sun Microsystems bejelentés és

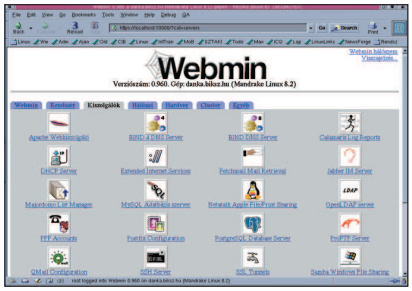


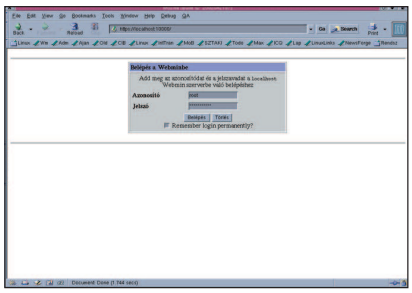
az azt követő StarOffice forráskódjának kiadása után alig telt el két esztendő, amikorra is az OpenOffice.org Project elkészítette végleges, minden igényt kielégítő irodai csomagját. A nyílt forrású irodai alkalmazás részét szövegszerkesztő, táblázatkezelő, bemutatókészítő, rajzoló és egyenletszerkesztő képezi.
A StarOffice 5.2 egyes elemeit eltávolították, például az összevont asztalt, a levelezőt, a naptárat és az ütemterv-kiszolgálót. Az irodai csomag nagymértékben együttműködik az MS Office-szal, így képes a legtöbb jelenleg elterjedt dokumentumtípus kezelésére, például a Microsoft Word 95/97/2000-esekére, az Excel 5.0/95/97/2000-esekére és a PowerPoint 97/2000-esekére is. A GPL felhasználási szerződésnek köszönhetően mind az otthoni, mind az irodai felhasználók tetszőleges számú számítógépen és teljesen ingyen használhatják. A program Linux, Solaris és Microsoft Windows 9x/NT/2000 operációs rendszereken futtatható. Az 1.0.0 változatszámmal rendelkező OpenOffice.org újdonságokat nem, viszont számos hibajavítást tartalmaz. CD-mellékletünkre a magyar helyesírás-ellenőrzővel (Myspell hu_HU 0.8) együtt került fel, mely, igaz, még nem teljes, azonban több kellemes meglepetést is tartogat a StarOffice 5.2-ről vagy a bármely más irodai alkalmazásról átállni kívánóknak. A helyesírás-ellenőrzőben végrehajtott hibajavításoknak köszönhetően szemtelenül gyors, míg a szótár tartalma rendkívül fejlett lett. Az ellenőrzés teljességét az OpenOffice.org biztosítja, mely a 641D változattól kezdve az összetett szavak írását is helyesen kezeli. A fent említett hibajavítások és újdonságok eredményeképpen az OpenOffice.org 1.0.0. irodai csomag és a hozzá tartozó Myspell helyesírás-ellenőrző telepítése minden felhasználó számára ajánlott.

Mozilla 1.0 RC1

Négyéves fejlesztés végeredményeként megjelent a Mozilla 1.0 Release Candidate változata. A Mozilla nyilvános forrású fejlesztői közössége az 1988-ban közzétett Netscape-forráskód alapján jött létre. A Mozilla Organization jelenleg az utolsó simításokat végzi a 1.0-s Release Candidate-változaton, mely a Mozilla.org oldaláról kipróbálás és véleményezés céljából letölthető. Annak ellenére, hogy a webböngésző kódját lezárták, több figyelemreméltó újdonsággal is találkozhatunk. A *Mozilla Composer* eszköz újdonsága a ***Publis Settings***, amely lehetővé teszi a szerkesztett HTML-dokumentum egyetlen gombnyomással történő mentését a webhely megfelelő könyvtárába. Helyet kapott a ***Download Manager***, mely a ***Tools*** menüpontból érhető el. A betűkezelés újdonsága a beállítható legkisebb betűméret, így többé nem kell mikroméretű betűtípusokkal találkoznunk a böngészés során. A fejlesztés jelenlegi állása szerint előreláthatólag még a Mozilla 1.1 július végi megjelenése előtt kiadásra kerül az 1.1 alfa- és a próbaváltozat. A böngésző a korongunkon megtalálható tar.gz és rpm-csomagok segítségével telepíthető.







Webmin

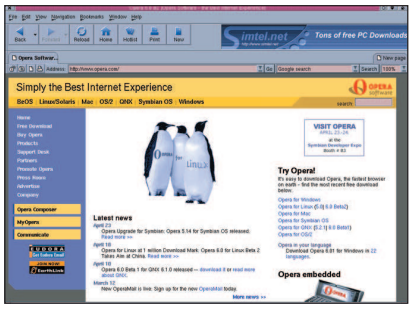
A Webmin kiválóan alkalmas a rendszergazdai teendők ellátására. Kiemelkedő sajátossága, hogy a Linuxszal még csak most ismerkedők számára is könnyen használható, emellett jól áttekinthető képet kapunk rendszerünkről és annak állapotáról. A háttérben futó Linuxot képes a maga nagyméretű és üzembiztos mivoltában megjeleníteni. A támogatott operációs rendszerek között szerepel a Red Hat, a Caldera, a SuSE, a Mand-

rake, a MSC Linux-, a Unix-változatok és a Solaris. Telepítése rendkívül egyszerű, akár az rpm, akár a tar.gz formátumot választjuk. Az első indítás a bármely böngészőnek megadott URL ➲ https://localhost:10000/ segítségével történik. A ***Webmin beállítása*** menüpont alatt változtathatunk az elérési címen, a kapucímen, valamint korlátozhatjuk azokat az IP-címeket, amelyekről engedélyezzük a Webminhez történő kapcsolódást. Használandó nyelvként érdemes a magyart választani, mivel csaknem a teljes kezelőfelület és a modulok is honosítva lettek. Ezután következhet rendszerünk beállítása, amit a ***Rendszer***, a ***Kiszolgálók***, a ***Hálózat*** és a ***Hardver*** menüpontok alatt végezhetünk el.

X-CD-Roast

Az X-CD-Roast az egyszerű CD-írást teszi lehetővé Linux alatt. Több parancssoros programot, mint a `cdrecord` és az `mkisofs` egyesíti egy jól kidolgozott grafikus felhasználói felület alatt. További előnye a CD-írók ***Burn-Proof*** támogatása, amely a feldolgozandó adatfolyam megszakadásából eredő hibás CD-k írását akadályozza meg. Ezenkívül a cdrecord egy belsőmemória-átmeneti tárral (FIFO) is rendelkezik, melynek méretét célszerű a CD-író átmeneti

táránál jóval nagyobbra állítani. Támogatja a ***DAO*** (tökéletes másolatot készít az eredeti CD-ről) és a ***TAO*** (a CD-t sávonként másolja 2 másodperces szünet közbeiktatásával), valamint a ***TAO zero pregap*** (sávonkénti másolás szünet nélkül) írási módokat. Képes olvasni a nyers-CD-k tulajdonságait, például a befogadóképességét, az ajánlott és a legmagasabb írási sebességét. Teljes körű CD-RW-támogatással rendelkezik, az írás mellett törölhető az egész CD vagy csak a tartalomjegyzék, egy sáv adatai, az egész sáv és az utolsó session adatai is. A fájlokat képes röptében vagy a sokkal biztonságosabb lemeznyomat (ISO) előállítása után felírni. Támogatja az ISO 9660 (DOS), a Rock Ridge (Unix) és a Joliet (Windows) fájlrendszert. A segítségével ezenkívül rendszerindításra alkalmas korongot (bootimage) hozhatunk létre, míg a több menetben írhatóra (multisession) készített lemezek írását bármikor folytathatjuk.

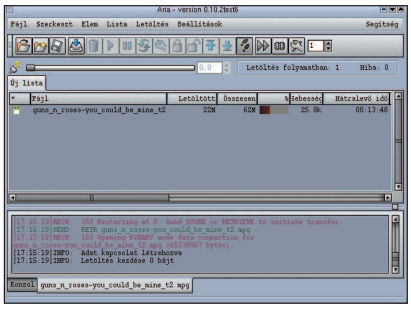


Opera

Elérhető az Opera 6.0 Linux második próbaváltozata. A böngésző szemlátomást gyorsabb lett, és a betűkezelés újdonságaként beállítható a legkisebb betűméret nagysága. A rengeteg hibajavításnak köszönhetően sokat javult a dokumentumkezelése, illetve a bővítmények támogatása. A jelenlegi változatban javítottak az ázsiai és a kínai nyelvek támogatottságán, így a végleges linuxos változattal remélhetőleg mind az ázsiai, mind a kínai weboldalak tökéletesen megjeleníthetőek lesznek. Az Opera 5 linuxos változatát eddig több mint egymillió alkalommal töltötték le, azonban az új nyelvi támogatással a norvég cég nem kis piac meghódítását vette célba, ami valószínűleg népszerűségének ugrásszerű növekedéséhez fog vezetni.

Apache 2.0.35

Megjelent a népszerű Apache Web Server régóta várt második nemzedékbeli, üzembiztos 2.0.35-ös változata. Az Apache Project által fejlesztett és karbantartott HTTP-kiszolgáló igen sokrétű, szabad forrású és ingyenes alkalmazás. A webkiszolgáló megjelenését a 2000 év végére tervezték, de az első próbaváltozata csak 2001 áprilisában jelent meg. Az új változat egy teljesen új programszerkezet alapján lett fejlesztve, amely



bizonyos helyzetekben lehetővé teszi a nagyobb teljesítmény elérését. A feladatok kiszolgálására ezentúl több lehetőség is a rendelkezésünkre áll: a végrehajtási szál (thread), amelynek kisebb az erőforrásigénye, és az eddig is alkalmazott folyamat- (process) kezelés. Beállítását több grafikus felülettel rendelkező eszközzel is elvégezhetjük. Választhatunk a ***Comanche***, a ***Mohawk***, a Linuxconf és a Webmin beállítóalkalmazások közül.

Aria

Már Linuxhoz is egyre több és jól használható letöltéskezelő programot készítenek. A legújabb fejlesztések eredményeképpen megjelent az Aria 0.10-es alkalmazás. Az igencsak alacsony változatszám senkit ne tévesszen meg – a program tud annyit, sőt többet is, amit divatos külsővel rendelkező társai. A program ismeri a HTTP-, a HTTPS- és az FTP-protokollokat. Képes proxykiszolgálóhoz kapcsolódni akár a felhasználónév és a jelszó megadásával, és párhuzamos letöltésekre és a megszakadt letöltések folytatására is alkalmas. Magyar nyelvi támogatással ugyancsak rendelkezik, így saját rendszerünk `LANG` beállításai alapján dönthetjük el, hogy magyar, illetve angol nyelven társalogjon-e. A letöltéskezelőktől nem megszokott, de a program tudását rendkívül hasznos kiegészítéssel látták el. Szerencsére egyre több kiszolgálón teszik lehetővé a letöltött anyag ellenőrzését a ***CRC*** vagy az ***MD5*** fájl segítségével. Az Aria ennek az ellenőrző számsornak a segítségével önműködően ellenőrizni tudja a letöltött anyagot, és képes jelezni azt is, ha letöltés közben megsérült. A program támogatja a húzd és dobd (Drag 'n Drop) módszert, így a letöltésre szánt fájlt egyszerűen csak be kell dobni az erre a célra kialakított kosárba.

Dankaházi István
(danka@linuxvilag.hu) a Linuxvilág munkatársa. Szabadidejében szívesen úszik és kerékpározik.

Tenyérnyi digitális fényképezőgép a Logitechtől

A Logitech Pocket Digital fényképezőgépet jobb, ha nem ejtjük be egy öblösebb táskába, mert nagyon nehéz lesz előkeríteni, miután megbújt valamelyik sarokban. A piciny, alumínium borítású csecsebecse a hitelkártyákkal

vállal méretbeli, a komolyabb gépekkel pedig tudásbeli rokonságot. CCD-je 1,3 megapixeles felbontásra képes, beépített 16 MB memóriájában a legjobb minőséget választva 52 db kép fér el. Áramellátásáról lítium-polimer akkumulátora gondoskodik, amely a számítógéppel létesített USB-csatlakozáskor azonnal fel is töltődik.
A 130 dolláros, azaz körülbelül 37 ezer forintos áron beszerezhető gép egyszerű, mindennapi használatra nagyszerű választásnak tűnik.

➲ http://www.logitech.com/cf/products/productoverview.cfm/4439

Távolról vezérelhető otthonok

Az Intel kutatói által közzétett fejlesztői készlet segítségével a programozók máris nekiláthatnak a háztartások távoli vezérlésére használható alkalmazások készítésének. A jövőben tehát a megfelelő mobiltelefonokról vagy zsebtitkárokról otthonunk működésébe is beleszólhatunk.

Az UPnP hálózatokba a személyi számítógépektől kezdve a legapróbb készülékekig bármi csatlakozhat, a lakás biztonsági rendszerétől egészen a házimozi berendezésig. Az így kiépülő otthoni hálózat természetesen arra is felhasználható, hogy otthon tárolt adatainkat, esetleg a digitális családi fényképalbumot a távolból nézegessük.
A fejlesztői készlet az Intel honlapjáról ingyenesen tölthető le.

➲ http://www.intel.com/pca/developernetwork/devsupport/index.htm

Elkészült az AC'97 kodek új változata

Az AC'97 kodek 2.3-as változatát olyan varázsszavak lengik körül, mint kétcsatornás digitális hangátvitel, 20 bites analóg-digitális és digitális-analóg átalakítók vagy a csatlakoztatott eszközök önműködő érzékelése. Utóbbi révén a számítógép felismeri, hogy illesztettünk-e valamilyen eszközt a hagyományos „jack" csatlakozóba , például mikrofont, illetve a hangszóró vagy fülhallgató impedanciáját is képes meghatározni. Az önműködő felismerés révén a hangkezelő alkalmazások, az internetes telefonálásra, hangjegyzetelésre vagy beszédfelismerésre használt programok továbbfejleszthetők. Az új szabvány pontos leírása pdf formátumban letölthető az Intel honlapjáról.

➲ http://www.intel.com/ial/scalableplatforms/audio/

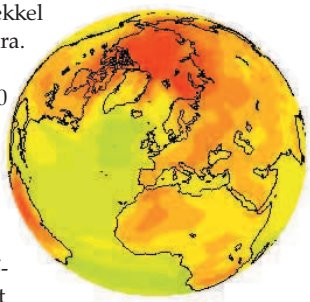
Visszatér a Trident

Hol vannak már a régi szép idők, amikor az volt a menő, akinek több memóriával – mondjuk 512 KB-tal – felszerelt Trident kártya volt a gépében! A Trident név gyakorlatilag évekre feledésbe merült, ám hamarosan megint jól fog csengeni. A Trident Microsystems és az UMC – a világ egyik legtekintélyesebb lapkagyártója – bemutatta az új, XP4 névre hallgató Trident grafikus lapkát. Az XP4 0,13 mikronos eljárással készül, teljes DirectX 8.1 támogatást biztosít, és másodpercenként egymilliárd képpont megjelenítésére képes. Magja 250, DDR memóriája pedig akár 666 MHz-es sebességgel ketyeg majd – miközben energiafelvétele nem éri el a 3 Wattot, amivel a legjobb teljesítmény/felvett energia arányt éri el.
A gyártók számára 40 dolláros áron elérhető lapka drágának sem mondható, így gyártásba kerülése után minden bizonnyal nagyon hamar feltűnik a hordozható gépekben.

➲ http://www.tridentmicro.com/

Mindenkiből lehet időjós

Az elosztott tervezetek nyomán haladva a Föld időjárási változásait vizsgáló tudósok hamarosan több ezer ember segítségét igénylik. A cél az, hogy a kutatók által felállított modelleket lefuttatva megállapítsák, vajon milyen klímaváltozásokra számíthatunk az elkövetkező ötven évben.
A létrehozott modellek közül minden résztvevőnek egyet kell majd lefuttatnia. Minden modell egyedi lesz abban az értelemben, hogy a többitől eltérő kezdeti feltételekkel indítják útjára.

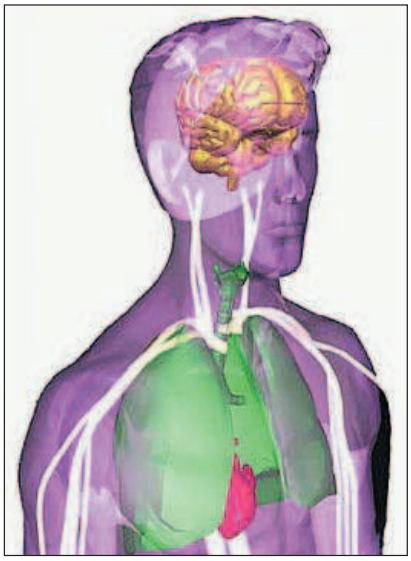


A modellek az 1950–2050 közötti időszakot fedik majd le, futásuk eredménye a meglehetősen összetett folyamatoknak megfelelően rendkívül eltérő lehet – a befutott eredmények közül azt fogják kiválasztani, amely az elmúlt több mint ötven év alapján a legjobban közelíti a jelenlegi állapotokat. A modellek természetesen összetettek, így a futtatásukat elvállalóknak fel kell készülniük arra, hogy számítógépük hosszú hónapokig nem fog unatkozni – kárpótlásul a modell életét egy interaktív szimulációval követhetik majd nyomon.
Az ügyfélprogram várhatóan nyár végére készül el.

➲ http://www.climateprediction.com

Linuxos szupergépet épít a HP

Az amerikai Energiaügyi Minisztérium Nyugati Parti Nemzeti Laboratóriuma (PNNL) és a HP bejelentették, hogy a PNNL egy 24,5 millió dolláros szuperszámítógép építésével bízta meg a HP-t.



A gépet tudományos számításokra fogják felhasználni, teljes – 2003-ra várható – üzembe helyezése után a világ legnagyobb Linuxot futtató számítógépe lesz. A szupergépbe az Intel Itanium processzorcsalád következő nemzedékének tagjaiból összesen 1400 fog kerülni, ezek összesített csúcsteljesítménye eléri majd a 8,3 teraflopot. A PNNL jelenleg használt gépét 1997-ben telepítették, és akkoriban a világ egyik legnagyobb teljesítményű számítógépe volt. Az új szupergép 1,8 TB memóriával és 170 TB lemezhellyel gazdálkodhat majd, így elődjénél harmincszor gyorsabb lesz, továbbá ötvenszer több tárhellyel és tízszer több memóriával fog rendelkezni.
➲ http://www.pnl.gov/

Grid Engine 5.3 a SuSe Linuxokban

A Sun Microsystems Grid Engine nevű rendszerének 5.3-as változata is belekerül a SuSe Linux 8.0-s Professional

Edition kiadásába. A Sun a bejelentés okán nem felejtette el méltatni saját, a Linux támogatásával kapcsolatos érdemeit – talán nem is ok nélkül –, állításuk szerint a cég e lépés révén egyike azoknak, akik a legtöbb szellemi értékkel támogatják a Linuxot és a nyílt forrású programok világát.
A Sun Grid Engine rendszere már most is több mint 4000 „grid" 150 ezer processzorán fut világszerte, ezek egyharmada Linux-környezetben működik.
A Grid Engine program 2000 szeptemberében jelent meg, hamarosan útjára indult a Grid Engine Project is, valamint szabadon elérhetővé vált közel félmillió sornyi programkód. A Grid Engine Project feladata, hogy a „grid"-megoldásokat továbbfejlessze, és használatukat a különféle géptípusokon elterjessze.
A rendszer immár számos operációs rendszer alá elérhető, így többek közt Solaris, Linux, Irix, Tru64, AIX és HP Unix alapon is futtatható.
A „gridben" kétféle gép juthat szerephez: mester és ügynök, ezek típusa akár eltérő is lehet. Maga a Grid Engine akár Windows-alapon is futhatna, azonban a Sun szerint erre nem volt még igény – a Solaris- és Linux-alapú gépek sokkal üzembiztosabbak.
➲ http://www.gridengine.sunsource.net

Fekete dobozt minden autóba

Az első halálos kimenetelű autós közúti baleset 1896-ban történt. Azóta mintegy harmincmillió ember lelte halálát az utakon, és rengetegen szenvedtek el valamilyen maradandó sérülést. A világon átlagosan minden percben meghal valaki.

A szomorú statisztikák arra sarkallják a gyártókat, hogy a lehető legbiztonságosabb járműveket építsék – ehhez azonban minél több adatot kell összegyűjteni, majd elemezni. Az IEEE P1616 számú tervezete a szakmai és amerikai kormányzati szakértők összefogásával olyan autós fekete dobozok szabványának kidolgozását célozza, amelyek a repülőgépekről megismert társaikhoz hasonló módon rögzítenék a gépjárművek menetadatait. A dobozok az elképzelések szerint a kocsik sebességét, irányát, az események idejét, a bent ülők számát és a biztonsági övek használatát jegyeznék fel egységes módon. A szabvány kiterjed majd az adatok formátumára és kinyerésük módjára is, a tényleges eszközök gyártása ez után kezdődhet majd meg.
➲ http://grouper.ieee.org/groups/1616/home.htm

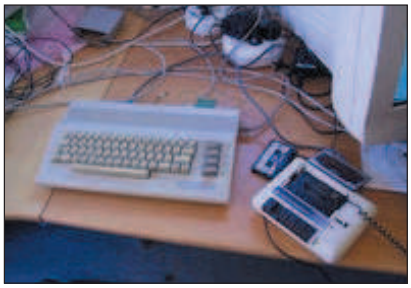
Újabb Creative Nomad Jukebox érkezett

A Creative Nomad sorozatának Jukebox tagjai eddig is kiemelkedtek a szürke mezőnyből, hiszen páratlan méretű tárhelyükkel és sokszínű szolgáltatásaikkal jóval többet nyújtanak, mint egy átlagos MP3-lejátszó. A frissített, 3. sorszámot viselő változatba 20 GB méretű merevlemez került, a folyamatos és energiatakarékos lejátszást 16 MB memória garantálja.

A készülék csatlakoztathatósága is bővült, rendelkezik infravörös, USB és ismeretlen okból SB1394-nek keresztelt Firewire csatlakozóval is. Az új Jukebox tudása, jellemzői roppant rokonszenvesek, egyedül 100 ezer forint körül alakuló ára törheti le tomboló vásárlási kedvünket.
➲ http://www.nomadworld.com/

Örökifjú C64

Két unatkozó fiatal már régóta foglalkozott azzal a gondolattal, hogy ethernet-csatolót kellene építeni Commodore 64-eshez. A gondolatot megfelelő hozzáértés hiányában csak jóval később követte tett, ám idővel mégis elkészült a



nagy mű. A csatolóhoz természetesen kellett egy IP-verem is – innen pedig már csak egyetlen lépés volt egy webkiszolgáló és egy RealAudio-kiszolgáló indítása.
A gép saját szalagos meghajtójáról szolgáltat zenét, üzemeltetői szerint is borzalmas minőségben, de kétségtelenül valós időben.
A készítők honlapján érdekes adatokat találhatunk munkájukról, és többek között az elkészült programok forráskódja is elérhető.
➲ http://dunkels.com/adam/tfe/

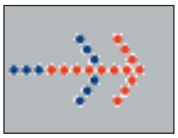
USB 2.0-s felületű szalagos meghajtó

Az OnStream új terméke az első olyan szalagos meghajtó, amely USB 2.0-s felületen csatlakozik a számítógéphez. Az ADR2.60usb névre keresztelt meghajtó tömörítéssel akár 60 GB adat mentésére is képes, a másolást legfeljebb 5 MB/s sebességgel tudjuk vele elvégezni. Az USB-felületű meghajtó legfontosabb előnye, hogy azok újraindítása és szerelés nélkül tetszés szerint mozgatható a gépek között. A meghajtóhoz a gyártó egy PCI-foglalatba illeszkedő USB 2.0-s illesztőkártyát is mellékel.

➲ http://www.onstreamdata.com

A világon elsőként indult MMS-szolgáltatás a Westelnél

A világ első kereskedelmi MMS szolgáltatását indította be április 18-án a Westel. A szolgáltatást minden előfizető és Domino kártyával rendelkező ügyfél külön előfizetési díj nélkül, egyszeri bejegyzés után igénybe veheti.



Az MMS szolgáltatás révén képeket és hangokat továbbíthatunk, a címzett a küldeményt elektronikus levélcímén vagy megfelelő mobiltelefonnal fogadhatja – ilyen készülék egyelőre egyetlen egy kapható hazánkban, a Sony-Ericsson T68i. MMS-t három változatban küldhetünk: a kis, legfeljebb 10 kbájt méretű üzenet díja 76 Ft + áfa, a közepes, 10–30 kbájt méretűé 160 Ft + áfa, a nagyméretűé pedig 320 Ft + áfa.

➲ http://www.westel900.net/elofizetes/szolgaltatasok/mms.html

Elkészült az OpenOffice.org 1.0

Május elseje nemcsak a munka, hanem az OpenOffice.org 1.0-s változat megjelenésének ünnepe is lett. A programcsomag elkészülte több mint másfél éves fejlesztőmunka eredménye. Amellett, hogy az OpenOffice.org ingyenes, fontos fegyvertény, hogy több nyelven is elérhető – közöttük magyarul is. A csomag Linux, Solaris és Windows alatt futtatható, és hosszas kínlódás árán végre a magyarországi csapatnak is sikerült lefordítania a honosított változatot Windows alá. Ezzel az utolsó akadály is elhárult az elől, hogy az OpenOffice.org meghódítsa a magyar felhasználók szívét is, használjanak akár Linuxot, akár Windowst.

OpenOffice.org 1.0
The Open Source Office Suite
AVAILABLE NOW!

➲ http://www.openoffice.hu

Támogatást vegyenek!

A Caldera bejelentette, hogy világszerte támogatást nyújt több népszerű Linux-terjesztéshez is, így a Red Hat, a SuSE, a Turbolinux, a Mandrake és a Conectiva Linuxokhoz. A cég 2000 augusztusában vásárolta fel az SCO-t, amely már működő támogatási részleggel rendelkezett.

CALDERA

Most e részleg tevékenységi körét bővítik, mivel a Caldera tapasztalatai szerint a felhasználók nem ragadnak le egyetlen terjesztésnél, hanem egy időben többféle Linuxot is használnak – ezentúl pedig egyetlen helyről kaphatnak segítséget mindegyik változathoz.

A rossz nyelvek szerint a támogatási részleg nem terjeszkedni akar, hanem kihasználtsági gondokkal küzd, ezért próbálnak új területekre lépni. Kérdéses egyrészt, hogy a saját támogatási szolgáltatással rendelkező Red Hat mennyire fog együttműködni, másrészt például egy Red Hat Linuxot vásárló ügyfél mennyire fog a Calderához fordulni támogatásért a Red Hat helyett.

➲ http://www.caldera.com/

Megszűnő Matáv-kedvezmények

Úgy tűnik, ezúttal nem alaptalan a pletyka: valóban megszűnnek a vezető magyar távközlési szolgáltató átalánydíjas telefonálási-internetezési csomagjai. Ahogy a cégnél fogalmaznak: „kivezetik őket a piacról". Július 1-jével már csak a múlté lesz a 150 forintosnak hívott kedvezmény valamint a Mindenkinek csomag.

Akinél műszakilag lehetőség van ADSL- vagy kábeltévés hozzáférés kiépítésére, valószínűleg nem nagyon fog bánkódni az eltűnő díjcsomagok miatt. Sajnos vannak azonban olyan honfitársaink is (nem kevesen), akik csak hagyományos telefonvonallal rendelkeznek, ők a Matáv „pótló" ajánlatai ellenére minden bizonnyal súlyos összegeket fognak fizetni, ha nem változtatnak internetezési szokásaikon.

➲ http://www.matav.hu/

Digitális alkotói verseny

A „Canon Digitális Alkotók Versenye – 2002" pályázat amatőr és profi művészek számára egyaránt lehetőséget nyújt a megmérettetésre. A 112 ezer amerikai dollár összdíjazású versenyen az alkotók négyféle tárgykörben adhatják be pályaműveiket: digitális kép, digitális grafika, illetve illusztráció, digitális film és végül web. A műveket szeptember 3-ig kell eljuttatni a megadott címek valamelyikére, az eredményhirdetés pedig decemberben lesz. A Canon pályázata tavaly is nagy sikert aratott, a 6209 mű a világ 57 országából futott be, közülük 7 magyar volt. Bővebb tájékoztatás kapható a

➲ http://www.canon.com/cdcc és a

➲ http://www.canon.hu címen.

Medgyesi Zoltán (mzx@axelero.hu) a BMGE 24 éves informatika szakos hallgatója. Szabadidejét legszívesebben a barátnőjével tölti. Szeret autózni és bográcsban főzni. A Linuxot hat éve ismeri, de még nem volt lelkiereje, hogy áttérjen rá. A Linuxvilág magazin hírszerkesztője.

Újra itt a nyár ...

A nyár közeledtével érdemes átgondolni, hogyan fogjuk megörökíteni a nyaralás élményeit, a tóparti marháskodásokat, a kerti szalonnasütéseket, és minden mást, amit érdemes. Olyan apró kis készüléket ajánlanék a számítógéphez kellően hozzánőtt olvasók számára, amely nemcsak nyáron, de a vacogós évszakokban is joggal kap helyet a zsebünkben vagy valahol a kezünk ügyében.

Egy átlagos kép – egy átlagos turista sem készít jobbat egy átlagos géppel

A PhotoClip a szó szoros értelmében tenyérnyi apróság, mérete felülről közelít egy családi gyufásdobozéhoz. Nem nevezhetem fényképezőgépnek, sem webkamerának, diktafonnak vagy MP3-lejátszónak, hiszen egyszerre alkalmas az összes feladatra, sőt, a digitális fényképezőgépektől megszokott módon mozgókép és sorozatkép rögzítésére is alkalmas. Öt az egyben, pici – mit is ér azonban mindez, ha nem tudjuk megfizetni? Nos, a PhotoClip bruttó 40 000 Ft körüli végfelhasználói ára viszonylag barátinak mondható. Míg egy komolyabb digitális fényképezőgép ára a kétszázezer forintot is elérheti, addig a PhotoClip bátran kivihető a szabadba, és ha esetleg valami baj történne vele – bár 40 000 Ft forint is nagy összeg –, mégsem egy fél éves magyar átlagfizetés vész kárba. Mit is kapunk ezért a pénzért? A csomagban a készülék mellett két AA típusú elem található, tovább egy CD-lemez néhány windowsos kiegészítő programmal, kézikönyvvel, hibaelhárítási útmutatóval, egy tetszetős fülhallgató, egy csuklószíj, egy apró, finom anyagú zacskószerű valami, amiben a gépet biztonságos(abb)an utaztathatjuk, valamint egy USB-kábel. Külön szeretném kiemelni a magyar nyelvű használati utasítást, amelynek létezése a hazai körülmények között már önmagában is szinte csoda.

A készülék olcsóságát egyfelől sajnos meg kell fizetni, hiszen lemaradt a TFT LCD kijelző, helyére csak egy egyszerű folyadékkristályos kijelző került, valamint a fényképek elkészítésére használt CCD érzékelő is csak 640×480-as felbontást tud felmutatni. Súlyos hátránya, hogy képeinket nem tudjuk azonnal visszanézni, tehát csak utólag, a számítógép képernyője előtt derül ki, ha valami nem úgy sült el, mint ahogy vártuk.

Másfelől viszont a készülék több olyan dolgot is tud, amelyek egy játékszer-kategóriás terméktől meglepőek. Először is bővíthető Compact flashkártyával, tehát nem muszáj a beépített 32 MB memóriára szorítkoznunk. Másrészt állványra tehető, amivel kezünk remegésének esetleges előnytelen hatásait küszöbölhetjük ki, valamint a beépített időzítőt segítségül hívva csoportképek készítésére is vetemedhetünk. Ha sötét helyen tartózkodunk, a gép beépített vakuja rövidebb távolságon belül megfelelően bevilágítja a teret, ha pedig szerzünk egy megfelelő adaptert, akkor külső áramforrásról is hajlandó táplálkozni, és nem kell felvásárolnunk a sarki közért elemkészletét. A fényképezéshez értők szívét talán megdobogtatja, hogy amíg a sokszor magasabb osztályba sorolható gépek nem képesek a fehéregyensúly állítására, addig a PhotoClip ötféle ilyen beállítást ismer, amelyek közül természetesen az egyik mód az önműködő. Érdekességszámba megy, hogy a gép elemtartó része elforgatható, és mivel az exponáló gomb is ennek tetejére került, kényelmetlen helyzetekben meglepő mutatványokat adhatunk vele elő.

A kezelőszervekkel gyorsan meg lehet barátkozni. Az üzemmódok között egy tekerőtárcsával választhatunk, az egyéb műveleteket pedig egy további nagyobb, kerek gombbal és négy kisebb nyomógombbal végezhetjük el. Ezek szerepe ugyan a pillanatnyi üzemmódtól függően módosulhat, ám az ésszerű működésnek és a feliratoknak köszönhetően nem nagyon jövünk zavarba – ha mégis, a leírás tömör és világos utasításokkal lát el bennünket. Lássuk egy kicsit részletesebben, hogy az egyes módokban mire is képes a PhotoClip!

Fényképezőgép

A 640×480-as felbontást teljes joggal „VGA- felbontásnak” nevezhetjük, ám ez mit sem változtat azon, hogy ez ez komoly célokhoz manapság már kevés. A képek készítésekor tapasztalatom szerint nem árt ügyelni néhány dologra. Például arra, hogy a gép nem tükörreflexes, és bár a kereső közel van a lencséhez, ha a gépet hanyagul fogjuk, a kezünk belelóghat a képbe. Az exponálás sem történik meg azonnal, a kép elkészültéről rövid pittyenés értesít – ha azonban ezt nem várjuk meg, akkor könnyen előfordulhat, hogy a gépet elrántjuk, és csak valami elfolyt folt kerül a memóriába.

A gép ötféle fehéregyensúlyt ismer: nappali fény, napsütés, neoncsöves fény, villanykörtefény és önműködő mód. A vakut hasonlóképpen kapcsolhatjuk önműködő és

kézimódba, valamint a képminőséget is két fokozatban állíthatjuk: a jó és a normál minőség közül választhatunk. Utóbbi esetben kétszerannyi kép fér a memóriába, állítólag a minőség gyengébb, bár nekem semmilyen különbséget nem sikerült felfedeznem a végeredmények között. Ha a módváltó tárcsát sorozatkép módba állítjuk, 1, 2 vagy 3 képből álló „sorozatlövést" is végezhetünk. Az önidőzítő 10 másodperc után exponál, ezen nem tudunk állítani. Tetszőleges hosszúságú megjegyzést fűzhetünk viszont minden képhez, mindössze annyi a dolgunk, hogy egy gombnyomással elindítsuk a felvételt.

…egy jobban sikerült kép

…és egy kevésbé sikeres mű

Diktafon

A jegyzeteléshez nagyon hasonló a diktafon mód, a különbség mindössze a létrejövő WAV-fájl elhelyezésében van. A hangjegyzetek a képek mellé kerülnek – a képekével megegyező sorszámmal –, míg a diktafon módban felvett anyagokat a gép memóriájában külön könyvtár várja.

MP3-lejátszó

MP3-lejátszóként a megszokott szolgáltatásokkal találkozunk: alapvető a hangerőszabályozás, a pillanatnyi és az összes szám ismételt lejátszása, továbbá a véletlenszerű lejátszás. Kevésbé mindennapi ötlet, hogy lejátszás közben jelölőket helyezhetünk el, a készülék később ezt a kijelölt szakaszt fogja ismételgetni. A hangszínszabályzót előre megadott stílusokhoz – pop, rock, jazz és vokális zenéhez – állíthatjuk be; külön állítgatásra, például mélyhangkiemelésre azonban sajnos nincs mód.

Mozgóképrögzítés

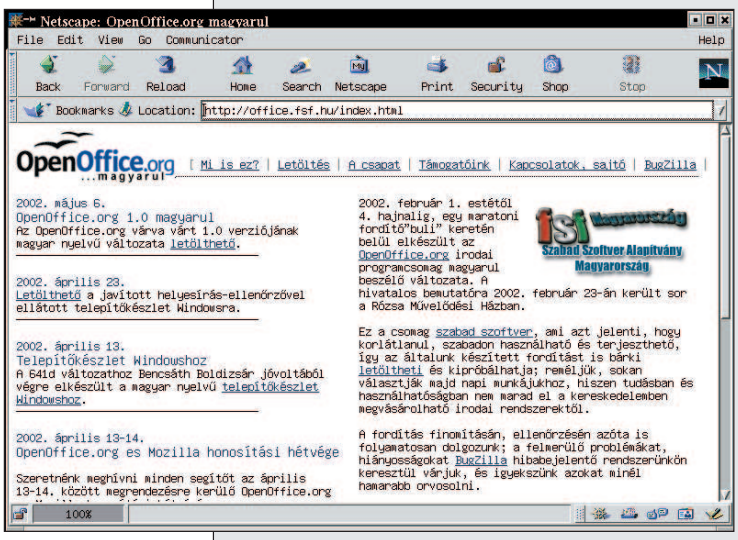
A PhotoClip már említett képességei alapján várható, hogy semmilyen komolyabb kamerát nem helyettesít. Már csak azért sem teheti, mert a flashmemóriák sebessége még túl kicsi ahhoz, hogy megfelelő minőségben lehessen rájuk mozgóképet rögzíteni.
Általában – tapasztalataim szerint – elmondható, hogy mozgókép ide vagy oda, a felvett kép lehetőleg minél kisebb része mozogjon, mert az eredmény elmosódott lesz.
Mozgókép módban is a kétféle minőség között választhatunk, jobb esetben 320×240 képpont, egyébként pedig 160×112 képpont lesz az MJPEG-tömörítéssel rögzített mozi felbontása. Az előbbi esetben 7, az utóbbiban pedig nagyjából 10 képkocka/másodperc képfrissítés jött össze, ami – ismerjük el őszintén – inkább csak a gyorsított diavetítés szintjét éri el.
Webkameraként nem próbáltam ki a készüléket. Általában a kifejezetten internetes célokra eladott piciny kamerákhoz valamilyen láb is tartozik, amellyel felhelyezhetők a képernyő tetejére, majd a tulajdonosukra állíthatók – a PhotoCliphez sajnos semmi ilyesmi nem tartozik, holott úgy gondolom, egy pár forintos műanyag lábbal meg lehetett volna oldani a dolgot.
A számítógéppel teremtett kapcsolat esetében leginkább azt tartom ésszerűnek, ha az USB-kábelt egyszerűen csatlakoztatjuk a gépekhez, majd a PhotoClipet USB-módba kapcsoljuk. Ekkor memóriájának tartalma három könyvtárba rendezve érhető el: a *DSCIMG* nevűbe kerülnek a fényképek, a sorozatképek, a filmek és a hangjegyzetek; a *Music for MP3* nevűbe az MP3-állományok, végül a gép a *VOICE* könyvtárba rögzíti a diktafon üzemmódban rögzítetteket. Windows alatt ehhez semmilyen különleges dolog nem kell, csatlakoztatás után a memória külön meghajtóként látszik, Linux alatt pedig csupán az `usb-storage` modul betöltését, majd a fájlrendszer befűzését kell elvégeznünk.
Mit lehetne rövid összefoglalásként elmondani a PhotoClipről? Lehetne egy jóízűt lovagolni gyengécske képminőségén, de ne feledjük el, hogy ugyanakkor szolgáltatásai szerteágazóak, és némi ügyességgel az alacsony felbontás ellenére is szép képeket készíthetünk vele. Fő érvként szolgál mellette alacsony ára, ami a vékonyabb pénztárcájú, vagy a fotózással csak alkalmi szórakozásként ismerkedők számára rendkívül vonzóvá teheti.
A PhotoClipnek létezik nagyobb memóriával, sőt, 1,3 Mpixel felbontású érzékelővel felszerelt változata is – vásárláskor tehát érdemes pontosan tájékozódni.
A készüléket az Alphasonic Kft. bocsátotta rendelkezésemre.

➲ http://www.alphasonic.hu
➲ http://www.worldisdigital.com

Medgyesi Zoltán
(mzx@axelero.hu) a BMGE 24 éves informatika szakos hallgatója. Szabadidejét legszívesebben a barátnőjével tölti. Szeret autózni és bográcsban főzni. A Linuxot hat éve ismeri, de még nem volt lelkiereje, hogy áttérjen rá. A Linuxvilág magazin hírszerkesztője.

Beszélgetés a Szabad Szofrverért Alapítvány alapítótagjaival

A hazai nyílt programok közösségében nemrég tűnt fel új és izgalmas kezdeményezéseivel a Szabad Szoftverért Alapítvány, más néven FSF.hu (Free Software Foundation Hungary). Ez a csoportosulás az 1985-ben megalakult nemzetközi FSF célkitűzéseit követve jött létre. Fő célja a Free Software projektek támogatása: az alapítók a GNU-alapú, illetve más szabadon hozzáférhető és módosítható forrású programok széles körben való elterjedését szeretnék elősegíteni.



A szervezet sokrétű feladatkör ellátására létesült: indulásakor felvállalta a felhasználói és fejlesztői érdekképviseletet, a tanácskozások szervezését, valamint a szabad programhasználat népszerűsítését a hazai médiában. Elsősorban azokkal az kezdeményezésekkel váltak ismertté a hazai közvélemény előtt, amelyek során egy-egy hosszú hétvégén, önként jelentkező résztvevők segítségével megszervezték a nyílt forráskódú OpenOffice.org programcsomag, majd a Mozilla böngésző felületének magyarítását. Az első honosítási hétvégére 2002. február 1–3 között, a másodikra április 13–14-én került sor. Mindkét rendezvény eredményesnek bizonyult.
Csak az OpenOffice magyarítása során 139 fordító működött közre, akik együttesen 21 348 kifejezést ültettek át angolról magyarra: majd ennek az óriási anyagnak az ellenőrzését, a kész anyag lektorálását is elvégezte az FSF.hu OpenOffice.org honosító csapata.
A jelenleg is folyamatban levő Magyar Mozilla Projekt során az FSF.hu a Mozilla program felhasználói felületének és súgójának fordításán dolgozik a bekapcsolódó szakemberek közreműködésével. A pillanatnyi frissítésekhez magyar nyelvi csomagot, valamint magyar nyelvű telepítőcsomagot készítenek és tesznek hozzáférhetővé.
Eddigi tevékenységükről kérdeztük a Szabad Szoftverért Alapítvány tagjait – természetesen elsősorban a fordítási munkálatokról, mely rövid fennállásuk során a legsikeresebb lépésük volt –, illetve jövőbeli terveikről érdeklődtünk.

Az alábbi interjú IRC formában zajlott le, hogy a szervezet mindegyik tagja válaszolhasson a Linuxvilág kérdéseire. Riportalanyaink ragaszkodtak ehhez a beszélgetési formához, ezzel is megtestesítve az FSF-ben uralkodó demokráciát.
Kérdéseinkre *Somogyi Péter* (Jerry), *Tímár András* és *Varga S. Csaba* (Guska) válaszolt, valamint *Bán Szabolcs* (Shooby), ő a csoport azon tagja, akit a kezdetekkor az FSF Europe felkért, hogy a magyar szabadprogram-mozgalom és a nemzetközi FSF között biztosítsa a kapcsolatot.

Szabó Ágnes: *Mióta létezik a szervezet, és hogyan alakult meg?*
Guska: Fél éve fogalmazódott meg először bennünk, hogy szükség van egy ilyen egyesületre, ekkor ültünk össze első ízben. Valójában mint baráti társaság jöttünk össze, és a mai napig nagyon fontos számunkra a baráti légkör. Az alapvető célkitűzés, amiért létrejöttünk, az volt, hogy a Linux érdekei mellett a szabad programok érdekei is képviselve legyenek Magyarországon. Egyelőre sajnos még nem működünk hivatalosan bejegyzett alapítványként, azonban legfőbb szándékunk, hogy hamarosan már ilyen formában tudjunk dolgozni. Úgy gondoltuk a kezdetekkor, hogyha az első feladatokat sikeresen elvégezzük, akkor érdemes jogi keretek közé terelni a tevékenységünket. Ennek mostanában jött el az ideje.
Sz. Á.: *Milyen kapcsolatban vagytok a nemzetközi FSF alapítvánnyal?*
Shooby: Az FSF.hu és az FSF Europe támogatásáról biztosított bennünket. Alapítványként való bejegyzésünk után fogjuk felvenni velük a hivatalos kapcsolatot.
Sz. Á.: Mi motivált benneteket, amikor megszerveztétek az OpenOffice.org és a Mozilla magyarításait?
Shooby: A szabad programok fejlesztése alapvetően a felhasználók igényein alapul. Szükség volt arra, hogy legyen magyar felülete is ezeknek a programcsomagoknak, és így széles körben használhatóvá váljanak.
Tímár András: A Mozillához ugyan volt magyar fordítás, de időközben elavult, ezért új változatra volt szükség.
Sz. Á.: *Hogyan sikerült erőforrásokat biztosítani a fordítási akciókra?*
Jerry: Több támogatónk volt, akiktől gépeket kaptunk. Az OpenOffice.org magyarításakor használhattuk a BME gépeit, hálózatát. Azonban nem kellett minden résztvevőnek gépet biztosítanunk, mivel sokan notebookkal érkeztek.
Shooby: Mind magánszemélyektől, mind cégektől nagyon sok hozzájárulást kaptunk eddigi munkánkhoz. Volt olyan, aki a magyarítási vállalkozásunk után utólag is támogatott bennünket.
Jerry: Bár az is igaz, hogy ezek a támogatások inkább eseti jellegűek voltak, nem nagyon fordult elő, hogy kétszer kaptunk volna anyagi segítséget ugyanattól.
Sz. Á.: *Hány embert mozgósítottatok a két fordítás alkalmával, és hogyan tudtátok mindezt megszervezni?*

Guska: Nagyon segítőkészek voltak a résztvevők, mindenki szívesen benne volt a munkában, így nem volt nehéz a szervezés. Az első fordítói hétvégét alig hirdettük meg. Mégis, mire első nap bementünk a BME-re, a nyersfordítás hat százalékát már elkészítették. Egyébként az OpenOffice.org és a Mozilla honosítása alatt összeállt egy tíz főből álló csapat, akik teljesen gördülékenyen tudnak együttdolgozni, és képesek irányítani a külső emberek munkáját is. Ez nagyon fontos számunkra a későbbi magyarítási feladatok szempontjából.

Jerry: Délután négy körül indítottuk el aznap (február 1-jén) a webes felületet, amelyről le lehetett tölteni a fordítandó anyagot. Anélkül, hogy ezen kívül bármit csináltunk volna, mire este hétkor elkészültünk a helyi hálózat összeállításával a BME-n, az anyag hat százaléka már le volt fordítva. A csapatmunkában egyébként az volt az érdekes, hogy a több mint száz ember között mindig akadt valaki, aki hozzá tudott szólni a különböző kérdésekhez. Felmerültek például kérdések nyomdászati szakkifejezésekkel, a táblázatkezelőben levő gazdasági függvényekkel kapcsolatban, és mindenre érkezett megoldás.

T. A.: Igen, csak nagy munka kijavítani, ha egy kérdésre rossz megoldás érkezik és az bekerül a kész változatba. Ezért szükséges később a többlépcsős lektorálás.

Sz. Á.: *A lefordított anyag lektorálását is sikerült már befejezni?*

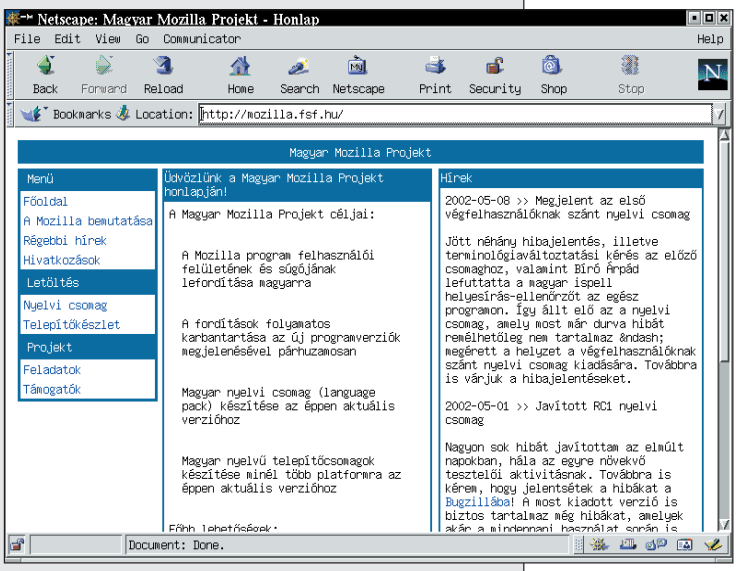
Jerry: Meg kell említenem, mert nem volt elég hangsúlyos az eddigi híradásokban, hogy a fordítás és lektorálás nemcsak a fordító összejöveteleken zajlik, hanem még azok után is hosszasan. A fordítás ellenőrzésén, finomításán a mai napig dolgozunk, ez mennyiségileg óriási munka, épp ezért több körben zajlik. Az OpenOffice.org fő lektora *Verók István*, a Mozillát Tímár András lektorálja *Bíró Árpád* és *Jánvári Gusztáv* közreműködésével.

Sz. Á.: *Tudjátok, hányan töltötték le eddig az oldalatokról a kész magyar változatokat?*

Jerry: Ezt pontosan nem tudjuk. Annyit tudunk, hogy a linux.index.hu-n egy szavazáson a magyar OpenOffice.org-ra körülbelül 400–500 szavazat érkezett. Továbbá elég sok levelet kaptunk akkor is, amikor másfél napig elérhetetlen volt a Mozilla nyelvi csomagja.

Sz. Á.: *Milyen terveitek vannak a továbbiakra? Készültök újabb magyarítási munkálatokra?*

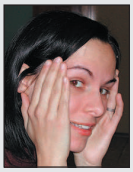
T. A.: Szeretnénk, bár nem sok olyan program van, amely akkora közérdeklődésre tarthat számot, mint az OpenOffice.org és a Mozilla. Felmerült például a Gimp honosítása. Nagy része ugyan már le van fordítva, de például a bővítmények honosítása még várat magára.



Jerry: A Gnome 2.0-s változatának fordítására is szükség lenne.

T. A.: Ez utóbbi azért is esélyes a magyarításra, mert a csomagjai elég jól meghatározottak, és egyszerre jön majd ki a program, nem pedig folyamatos fejlesztésekkel, amelyeket a magyar felület változtatásával követnünk kellene.

Shooby: A jövőben egyébként nemcsak fordítással kívánunk foglalkozni, hanem tevékenységünket ki szeretnénk terjeszteni a többi, általunk vállalt feladatra is.



Szabó Ágnes
(agi@netelligents.com) újságíró, szociológus. 25 éves, sajtó-, internet- és kávéfüggő. Ez utóbbiról szerencsére leszokófélben van. Kedvenc sportja a futás és a lovaglás.

A következő lépés: Linuxszal, ugyanennyiért

Szeretnéd tudni, mennyibe kerül egy névtelen PC önmagában? A legjobb hely, ahol kiderítheted, a Wal-Mart. A Wal-Mart weboldalán találhatsz általános, 1 GHz Intel processzoros névtelen gépet (Microtel SYSMAR116) a manapság szokásos jellemzőkkel és számokkal: 128 MB RAM, 40 GB merevlemez, billentyűzet, egér, modem, aktív hangfal, hajlékonylemez stb. Továbbá nagybetűkkel fogad téged a köszöntő:

MONITOR ÉS OPERÁCIÓS RENDSZER NÉLKÜL
Van még hely, ahol az árucikkeket áruként kezelik.
Ára: 399 dollár
A weboldal helye:
➲ www.walmart.com/catalog/product.gsp?product_id=1731327

Doc Searls

Guska: Nagyon segítőkészek voltak a résztvevők, mindenki szívesen benne volt a munkában, így nem volt nehéz a szervezés. Az első fordítói hétvégét alig hirdettük meg. Mégis, mire első nap bementünk a BME-re, a nyersfordítás hat százalékát már elkészítették. Egyébként az OpenOffice.org és a Mozilla honosítása alatt összeállt egy tíz főből álló csapat, akik teljesen gördülékenyen tudnak együttdolgozni, és képesek irányítani a külső emberek munkáját is. Ez nagyon fontos számunkra a későbbi magyarítási feladatok szempontjából.

Jerry: Délután négy körül indítottuk el aznap (február 1-jén) a webes felületet, amelyről le lehetett tölteni a fordítandó anyagot. Anélkül, hogy ezen kívül bármit csináltunk volna, mire este hétkor elkészültünk a helyi hálózat összeállításával a BME-n, az anyag hat százaléka már le volt fordítva. A csapatmunkában egyébként az volt az érdekes, hogy a több mint száz ember között mindig akadt valaki, aki hozzá tudott szólni a különböző kérdésekhez. Felmerültek például kérdések nyomdászati szakkifejezésekkel, a táblázatkezelőben levő gazdasági függvényekkel kapcsolatban, és mindenre érkezett megoldás.

T. A.: Igen, csak nagy munka kijavítani, ha egy kérdésre rossz megoldás érkezik és az bekerül a kész változatba. Ezért szükséges később a többlépcsős lektorálás.

Sz. Á.: *A lefordított anyag lektorálását is sikerült már befejezni?*

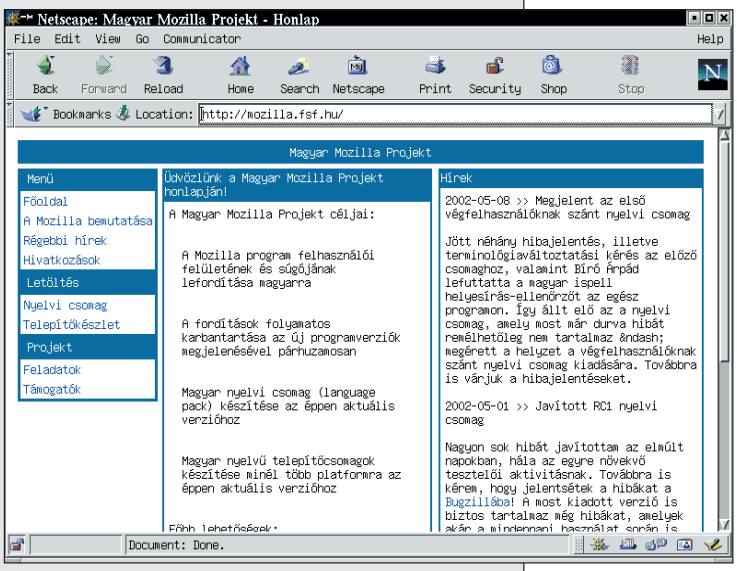
Jerry: Meg kell említenem, mert nem volt elég hangsúlyos az eddigi híradásokban, hogy a fordítás és lektorálás nemcsak a fordító összejöveteleken zajlik, hanem még azok után is hosszasan. A fordítás ellenőrzésén, finomításán a mai napig dolgozunk, ez mennyiségileg óriási munka, épp ezért több körben zajlik. Az OpenOffice.org fő lektora *Verók István*, a Mozillát Tímár András lektorálja *Bíró Árpád* és *Jánvári Gusztáv* közreműködésével.

Sz. Á.: *Tudjátok, hányan töltötték le eddig az oldalatokról a kész magyar változatokat?*

Jerry: Ezt pontosan nem tudjuk. Annyit tudunk, hogy a linux.index.hu-n egy szavazáson a magyar OpenOffice.org-ra körülbelül 400–500 szavazat érkezett. Továbbá elég sok levelet kaptunk akkor is, amikor másfél napig elérhetetlen volt a Mozilla nyelvi csomagja.

Sz. Á.: *Milyen terveitek vannak a továbbiakra? Készültök újabb magyarítási munkálatokra?*

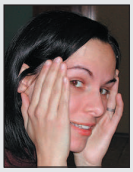
T. A.: Szeretnénk, bár nem sok olyan program van, amely akkora közérdeklődésre tarthat számot, mint az OpenOffice.org és a Mozilla. Felmerült például a Gimp honosítása. Nagy része ugyan már le van fordítva, de például a bővítmények honosítása még várat magára.



Jerry: A Gnome 2.0-s változatának fordítására is szükség lenne.

T. A.: Ez utóbbi azért is esélyes a magyarításra, mert a csomagjai elég jól meghatározottak, és egyszerre jön majd ki a program, nem pedig folyamatos fejlesztésekkel, amelyeket a magyar felület változtatásával követnünk kellene.

Shooby: A jövőben egyébként nemcsak fordítással kívánunk foglalkozni, hanem tevékenységünket ki szeretnénk terjeszteni a többi, általunk vállalt feladatra is.



Szabó Ágnes
(agi@netelligents.com) újságíró, szociológus. 25 éves, sajtó-, internet- és kávéfüggő. Ez utóbbiról szerencsére leszokófélben van. Kedvenc sportja a futás és a lovaglás.

A következő lépés: Linuxszal, ugyanennyiért

Szeretnéd tudni, mennyibe kerül egy névtelen PC önmagában? A legjobb hely, ahol kiderítheted, a Wal-Mart. A Wal-Mart weboldalán találhatsz általános, 1 GHz Intel processzoros névtelen gépet (Microtel SYSMAR116) a manapság szokásos jellemzőkkel és számokkal: 128 MB RAM, 40 GB merevlemez, billentyűzet, egér, modem, aktív hangfal, hajlékonylemez stb. Továbbá nagybetűkkel fogad téged a köszöntő:

MONITOR ÉS OPERÁCIÓS RENDSZER NÉLKÜL
Van még hely, ahol az árucikkeket áruként kezelik.
Ára: 399 dollár
A weboldal helye:
➲ www.walmart.com/catalog/product.gsp?product_id=1731327

Doc Searls

Flexlm, az elektronikus vagyonőr

A Flexlm egy olyan kereskedelmi licencszolgáltató programcsomag, ami nem csak Linuxon fut. Nagy az esély rá, hogy előbb-utóbb beleütközünk, még akkor is, ha a saját gépünkön csakis szabad programok találhatók.

A Flexlm olyannyira kereskedelmi program, hogy még a mindenki által letölthető Flexlm Végfelhasználói Útmutató tartalma is bizalmas adatnak számít, ezért mostani írásomban nagy erőfeszítéseket tettem arra, hogy úgy ismertessem a programcsomagot, mintha semmit sem tudnék róla. Bizalmasan mégis közlöm, hogy ismereteimet nem légből kaptam.

1. lista Az ifconfig eth0 parancs kimenete

```
eth0      Link encap:Ethernet  HWaddr 00:90:1F:5A:00:A5
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:10 Base address:0xa000
```

A programcsomagot a kaliforniai Globetrotter Software Inc. fejleszti (lásd a ➲ http://www.globetrotter.com honlapot). Feltételezem, hogy a Flexlm név a flexible license manager (rugalmas engedélyintéző) szavak rövidítéséből keletkezett, de lebegő, sodródó felhasználási szerződés (floating license) szolgáltatásról is beszélhetünk, mivel a programcsomag segítségével az engedély bárhová „átúszhat" a hálózaton keresztül. Az egésznek a szíve a `lmgrd` démon, ami az engedélyszolgáltató kiszolgálón tartja a kapcsolatot a tulajdonukat védeni akaró cégek ügyfélgépein futó kereskedelmi démonokkal (vendor daemon). Tudjuk, hogy a démonok olyan folyamatok, amelyek a háttérben futnak és nem interaktívak, azaz semmilyen felhasználói beavatkozásra nem várnak. Általában valamilyen szolgáltatást nyújtanak a rendszer egészének vagy egy felhasználói programnak. A démonok rendszerindításkor indulnak, és a rendszer leállításáig futnak, hacsak valamilyen okból időközben le nem állítjuk őket. Esetünkben a kereskedelmi démonok tartják számon, hogy hány felhasználónak van engedélye az adott program használatára, és hogy kik ezek a felhasználók, valamint ők tartják a kapcsolatot az ügyfélalkalmazásokkal egy bármilyen összetételű hálózaton keresztül, ami akár az Internet is lehet. Ha ezek közül a kereskedelmi démonok közül bármelyik leáll, az adott felhasználócsoport elveszti az engedélyét. A felhasználói programok rendszeres időközönként felveszik a kapcsolatot az engedélyszolgáltató démonnal, amiről egy az alkalmazáshoz csatolt Flexlm-programrész gondoskodik.
A végfelhasználói programok feltelepítésük után kapcsolatot teremtenek az `lmgrd` démonnal, ami közli velük a kereskedelmi démon hálózati címét. Ezután a megkeresett kereskedelmi démon az adatbázisában ellenőrzi, hogy az ügyfélgépnek van-e jogosultsága az adott program futtatására, és ettől függően engedélyezi vagy megtagadja a program használatát. Ha a felhasználásra irányuló kérés jogos, akkor a kereskedelmi démon a végfelhasználó számára felhasználási szerződésfájlt bocsát ki, ami adatokat tartalmaz az engedélyezendő termékről, annak telepítési helyéről, az engedély időtartamáról, esetlegesen az engedélyszolgáltató és a kereskedelmi démonokról. Miután a felhasználási szerződés fájlja megérkezik az ügyfélgéphez, az engedélyt kérő programba beépített Flexlm-modul engedélyezi az alkalmazás elindítását, feltéve, hogy mindent rendben talál. Lehetőség van arra is, hogy a programot egyetlen géphez kössük. Ilyen esetben nincs szükség az engedélyszolgáltató és a kereskedelmi démon jelenlétére, és a cégtől kapott felhasználási szerződésfájlt az `installKey` grafikus felülettel rendelkező programmal telepíthetjük. Egyes felhasználóknak többet engedélyezhetünk, másokat pedig eltilthatunk bizonyos lehetőségek használatától.
A programcsomagban több hasznos felhasználási szerződés felügyeleti segédeszköz is található:

- Az ***lmborow*** program segítségével felhasználási szerződést kölcsönözhetünk a parancssoron megadott kereskedelmi démontól a megadott időpontig, akár másodperces pontossággal is.
- Az ***lmdiag*** programot akkor kell elővennünk, ha a felhasználási szerződés lekérésekor valamilyen problémánk adódik.
- Az ***lmdown*** felügyeleti segédeszköz leállítja a felhasználásiszerződés-démonokat, egyszerre az összeset vagy csak egyet, esetleg több megadottat.
- Az ***lminstall*** olvasható formátumú felhasználási szerződésfájlt hoz létre, amit könnyebb begépelni. A másik fajta felhasználási szerződésfájl az úgynevezett tizedes formát tartalmazza, ami betűk és számok egymás mellé hányt sorozata.
- Az ***lmnewlog***, az *lmswitch* és az *lmswitchr* alkalmazások segítenek a naplózó és hibajelentő naplófájlok felügyeletében, lezárnak egy létező naplófájlt, majd egy újat nyitnak meg akár a fő felhasználásiszerződés-szolgáltató kiszolgálógépen, akár a kereskedelmi démonok gépein.
- Az ***lmpath*** segédeszköz a Flexlm felhasználásiszerződés-elérési utak felügyelését segíti. Például új elérési utakat adhatunk hozzá a már meglévő elérésiút-állományhoz, vagy felülírhatjuk azokat.
- Az ***lmremove*** program eltávolítja a távoli felhasználói engedélyt, amikor egy ügyfélgép operációs rendszere összeomlik, mivel ilyenkor ez nem történik meg önműködően. A rendszer visszaállítása után új felhasználási szerződésért kell folyamodni.

- Az ***lmreread*** program újraolvassa a felhasználási szerződésfájlokat, ha az engedélyezésben valami változás történik, és figyelmezteti a többi futó démont is, hogy tegye meg ugyanezt.
- Az ***lmstat*** eszköz segít nyomon követni a hálózati felhasználásiszerződés-engedélyezési folyamatokat, szemmel tartja a futó démonokat, a felhasználókat, az egyedi, különleges kívánságokat kiszolgáló kereskedelmi démonokat és a felhasználásiszerződés-kölcsönzések helyzetét.
- Az ***lmver*** kiírja, hogy az adott gépen melyik változatszámú Flexlm-csomag található, például az `lmver lmutil` parancs kimenete a következő:

```
lmver - Copyright (C) 1989-1999
Globetrotter Software, Inc.
FLEXlm v7.0d (liblmgr.a), Copyright
(C) 1988-1999 Globetrotter Software,
Inc.
```

Megjegyzem, a Flexlm programcsomag jelenleg már a 8.0-s változatnál tart, és támogatja a Windows különböző fajtáit, mint például a Windows 95/98-at, a Windows NT-t, a Windows 2000-et, a Windows ME-t és a Windows XP-t. Több Unix-változaton fut, többek közt a Linuxon is, de úgy tűnik, hogy jelenleg a Windowsra írt változat a jobban kidolgozott és több szolgáltatás nyújt. Ez azt hiszem, nem túl meglepő.

- Az ***lmhostid*** segítőalkalmazás a gép Flexlm-azonosítóját írja ki a parancssora. Ez az azonosító életbevágóan fontos a felhasználásiszerződés-szolgáltató szempontjából, hiszen neki képesnek kell lennie arra, hogy egyértelműen azonosítani tudja azt az ügyfélgépet, amelyik engedélyezési kéréssel fordult hozzá. Bizonyos számítógépek rendelkeznek beépített azonosítási lehetőséggel, mint például a Sun Microsystems gépei, de az átlagos munkaállomások nem mindig azonosíthatók egyértelműen. Ilyenkor azonosításra kézenfekvő a hálózati kártyát használni, hiszen annak mindig egyedi száma van. Jogos feltevés az is, hogy a gépben már van hálózati kártya, hiszen a felhasználásiszerződés-szolgáltatás a hálózaton keresztül történik. A Flexlm azonosító a hálózati kártya számából egyszerűen a következő logika szerint jön létre. Például a `/sbin/ifconfig eth0` paranccsal írassuk ki a hálózati kártya számát (a kimenetet lásd az *1. listán)*.

A kártya azonosítója a `HWaddr` rövidítés utáni szám, ami a példában `00:90:1F:5A:00:A5`. A Flexlm gépazonosító egyszerűen adódik a fenti számból, ha a kettőspontokat elhagyjuk. De még a hálózati kártya címét sem kell megnéznünk, ha az *lmhostid* parancssori segédeszközt használjuk, mert ez azonnal kiírja nekünk gépünk Flexlm-azonosítóját. Például:

```
lmhostid - Copyright (C) 1989-1999
Globetrotter Software, Inc.
The FLEXlm host ID of this machine is
"00901f5a00a5"
```

Egy szó eltéréssel ugyanezt a kimenetet kapjuk, ha az `lmutil lmhostid` parancsot adjuk ki:

```
lmutil - Copyright (C) 1989-1999
Globetrotter Software, Inc.
The FLEXlm host ID of this machine is
"00901f5a00a5"
```

Ezen nem kell csodálkozni, hiszen az összes segédeszköz az *lmutil* alkalmazásba van csomagolva, és a fent említett programok nevei nem mások, mint közvetett hivatkozások az *lmutil*-hez. Amikor a parancssoron azt írjuk be, hogy `lmutil lmhostid`, akkor tulajdonképpen az `lmhostid` parancsot hívjuk meg (lásd a *2. listát*).

2. lista Az lmhostid parancs kimenete

```
lrwxrwxrwx    1 root     root            6 Apr 10 10:26 lmcksum -> lmutil
lrwxrwxrwx    1 root     root            6 Apr 10 10:26 lmdiag -> lmutil
lrwxrwxrwx    1 root     root            6 Apr 10 10:26 lmdown -> lmutil
lrwxrwxrwx    1 root     root            6 Apr 10 10:26 lmhostid -> lmutil
lrwxrwxrwx    1 root     root            6 Apr 10 10:26 lmremove -> lmutil
lrwxrwxrwx    1 root     root            6 Apr 10 10:26 lmreread -> lmutil
lrwxrwxrwx    1 root     root            6 Apr 10 10:26 lmstat -> lmutil
-rwxr-xr-x    1 root     root       312220 Jan 16  2001 lmutil
lrwxrwxrwx    1 root     root            6 Apr 10 10:26 lmver -> lmutil
```

A Linuxban mindig a hálózati kártya száma szolgál az azonosítás alapjául, de más operációs rendszerekben lehetőség van a felhasználási szerződést például a merevlemez sorozatszámához, a processzorhoz, a párhuzamos vagy USB-kapura csatlakoztatott hardverkulcshoz, IP-címhez, sőt akár egy monitorhoz is kötni.

Korszerű világunk azonban nemcsak a hálózatokról szól, hanem a hordozható, ölbe vehető számítógépekről is. Ez utóbbiak természetüknél fogva nem mindig lógnak a hálón, tulajdonosuk időnként leválasztja őket, és magával viszi otthonába vagy vidéki útjára. Néhány szorgalmas laptoptulajdonos még ilyenkor is használná kedvenc programját, ha azt a Flexlm nem tenné használhatatlanná, amely a hálózatról való leválasztás után nem érzékeli többé a felhasználásiszerződés-szolgáltató jelenlétét, és emiatt lehetetlenné teszi a program futtatását. Megoldás lehet ilyenkor a felhasználásiszerződés-szolgáltatók ellenőrző szerepének kiiktatása, és a felhasználási szerződésfájl kihelyezése az adott gépre. A felhasználási szerződés ilyenkor

a mozgatható számítógépekhez kötődik, és a program csak azon az egyetlen gépen lesz hajlandó elindulni, ahol a felhasználási szerződésfájl található.
Hasonló nehézség adódik olyankor is, amikor egy személy két munkaállomáson dolgozik, de nem egy időben. Az egyiken munkaidőben a munkahelyén, a másikon az otthonában a szabadidejében, ahol tovább folytatja addigi munkáját. A felhasználó szempontjából egyértelmű a helyzet, hiszen ugyanazt a munkafolyamatot végzi tovább ugyanazon programmal, csak más helyen.
Az engedélyezési eljárás azonban nem a személyéhez kötődik, hanem a munkahelyi géphez. Hiába tehát az eszményi munkaerő szorgalma, ha Flexlm az engedélyt csak a munkahelyi gép számára hajlandó megadni, az otthoninak nem. Hiába van rajta mindkét gép a hálózaton, hiába látják a felhasználásiszerződés-szolgáltatót, ha csak egyiküket hajlandó kiszolgálni. A fenti esetekre találták ki a mozgatható (mobile) felhasználási szerződést. Képzeljünk el egy kis szerkezetet a párhuzamos vagy az USB-kapura csatlakoztatva, amihez külön meghajtót kell telepíteni, és ami egy úgynevezett Flexid-azonosítót szolgáltat. Ilyenkor elegendő kihúzni a szerkezetet és bedugni a másik gépbe, hogy az ott lévő program futtathatóvá váljon. Mivel csak egy ilyen szerkezetünk van, az adott programot egy időben csak egy gépen lehet futtatni. Ilyenkor mindkét gépre vagy akár több gépre ugyanaz a felhasználási szerződésfájl van átmásolva, de mindegyik ugyanazon Flexid-azonosítóra vár.
Sokat hallunk mostanában hasonló Microsoft-próbálkozásokról is. Például a Windows XP egyes változatait bizonyos idő eltelte után az Internet segítségével aktiválni kell. Feltehető, bár a forráskód titkossága miatt tudni nem szabad, hogy az XP esetében is a fent leírt eljárásokhoz hasonlókat alkalmaznak. Igen valószínű az is, hogy a Microsoft által használt megoldás alapgondolata nem a cég programmérnökeinek fejéből pattant ki. Ahogy eddig számtalanszor megtörtént, a tőkeerős cég ismét csak lemásolta mások ötleteit, és létrehozta saját, együttműködésre képtelen változatát, hogy kiszorítsa a piacról az eredeti alkotókat. Mivel nem tudjuk, hogy a Microsoft miképpen azonosítja gépünket, jobb, ha azt feltételezzük, hogy az XP és társai nem egyszerűen a számítógépünkben lévő alkatrészek jellemzőiből, például a merevlemez és a memória méretének átlagolásából nyernek ki valamilyen, a hálózaton a cég központjába küldendő bűvös számokat, hanem igencsak megfogható adatok után kutatnak gépünkön, például a hálózati kártya egyedi száma után.
Eme elrettentő példa ellenére sem tartom rossz ötletnek a Flexlm által nyújtott megoldásokat. Eljöhet az idő, amikor a maihoz képest igen gyors hálózatokon majd különböző programcsomagokhoz férhetünk hozzá, amelyek használatát egy máshol lévő felhasználásiszerződés-szolgáltató fogja engedélyezni, és mi csak annyit fogunk fizetni a programok futtatásáért, amennyi ideig ténylegesen használtuk azokat. Olyan lenne ez, mint manapság a telefonhasználat. A telefonszámlán csak a tényleges beszélgetések ellenértékének összege jelenik meg.
Ehhez természetesen nemcsak gyors hálózatra lenne szükség, hanem kiegyensúlyozott programpiacra, hiszen nem tekinthető normális helyzetnek az, hogy ugyanannak a programnak az ára nullától akár több millió forintig terjedhet. Példaként említhetem az Oracle adatbáziskezelőt, aminek Linuxra írt változatát ingyenesen megkaphattuk otthoni tanulmányozásra, holott piaci ára ennek is a milliót közelíti. Számos nagyon jó program elterjedésének éppen magas ára szab határt. Lehetséges tehát, hogy a Flexlmhez hasonló megoldások a jövőben nemhogy korlátozni fogják a programok terjedését, hanem éppenséggel elősegíthetik, ha segítségükkel leszoríthatjuk az árakat.

Szaló István (ratiosoft@freemail.hu) tanár, immár több mint másfél évtizede foglalkozik programozással, de csak a Java és a Linux megismerése után tudta meg, hogy mi is az igazi programozás. Több írása megjelent már a hazai számítástechnikai lapokban.

Netcraft – és a hajó megy

A Netcraft legutóbbi (2002. januári), webkiszolgálókról készült felmérése szerint változatlanul az Apache-é a működő weboldalak 63,69 százaléka, ami az előző eredményéhez képest 0,35 százalékos növekedést jelent. A Microsoft IIS részesedése 0,55 százalékkal csökkent, jelenleg 26,07 százalékon áll. Mindkettő esetében nőtt a kiszolgálók száma. Az iPlanet a harmadik 2,99 százalék részesedéssel, a negyedik helyen pedig a Zeus áll 2,16 százalékkal – lényegében mindkettő változatlan maradt.
A Netcraft-jelentés a Sun Cobalt leányvállalatát illetően – amely Linux-alapú webkiszolgálókat forgalmaz – jót és rosszat is tartalmazott. „Habár az IP-címek száma a Cobalt-gépeken az elmúlt évben emelkedett, részesedésük a Linuxon futó összes weboldal tekintetében mégis csökkent, jóformán folyamatosan, hónapról hónapra." A Netcraft azt is megjegyezte, hogy Cobaltról két nagy ügyfél is hagyományos Linux gépekre váltott át. A texasi Everyone's Internet internetszolgáltató vállalat nemrég bejelentette „a legnagyobb értékű beszerzési megállapodást, amit független, észak-amerikai internetszolgáltató a Cobalttal valaha is kötött." A vállalat hétszáz Cobalt RaQ-kiszolgálót vásárolt.

Doc Searls

a mozgatható számítógépekhez kötődik, és a program csak azon az egyetlen gépen lesz hajlandó elindulni, ahol a felhasználási szerződésfájl található.
Hasonló nehézség adódik olyankor is, amikor egy személy két munkaállomáson dolgozik, de nem egy időben. Az egyiken munkaidőben a munkahelyén, a másikon az otthonában a szabadidejében, ahol tovább folytatja addigi munkáját. A felhasználó szempontjából egyértelmű a helyzet, hiszen ugyanazt a munkafolyamatot végzi tovább ugyanazon programmal, csak más helyen.
Az engedélyezési eljárás azonban nem a személyéhez kötődik, hanem a munkahelyi géphez. Hiába tehát az eszményi munkaerő szorgalma, ha Flexlm az engedélyt csak a munkahelyi gép számára hajlandó megadni, az otthoninak nem. Hiába van rajta mindkét gép a hálózaton, hiába látják a felhasználásiszerződés-szolgáltatót, ha csak egyiküket hajlandó kiszolgálni. A fenti esetekre találták ki a mozgatható (mobile) felhasználási szerződést. Képzeljünk el egy kis szerkezetet a párhuzamos vagy az USB-kapura csatlakoztatva, amihez külön meghajtót kell telepíteni, és ami egy úgynevezett Flexid-azonosítót szolgáltat. Ilyenkor elegendő kihúzni a szerkezetet és bedugni a másik gépbe, hogy az ott lévő program futtathatóvá váljon. Mivel csak egy ilyen szerkezetünk van, az adott programot egy időben csak egy gépen lehet futtatni. Ilyenkor mindkét gépre vagy akár több gépre ugyanaz a felhasználási szerződésfájl van átmásolva, de mindegyik ugyanazon Flexid-azonosítóra vár.
Sokat hallunk mostanában hasonló Microsoft-próbálkozásokról is. Például a Windows XP egyes változatait bizonyos idő eltelte után az Internet segítségével aktiválni kell. Feltehető, bár a forráskód titkossága miatt tudni nem szabad, hogy az XP esetében is a fent leírt eljárásokhoz hasonlókat alkalmaznak. Igen valószínű az is, hogy a Microsoft által használt megoldás alapgondolata nem a cég programmérnökeinek fejéből pattant ki. Ahogy eddig számtalanszor megtörtént, a tőkeerős cég ismét csak lemásolta mások ötleteit, és létrehozta saját, együttműködésre képtelen változatát, hogy kiszorítsa a piacról az eredeti alkotókat. Mivel nem tudjuk, hogy a Microsoft miképpen azonosítja gépünket, jobb, ha azt feltételezzük, hogy az XP és társai nem egyszerűen a számítógépünkben lévő alkatrészek jellemzőiből, például a merevlemez és a memória méretének átlagolásából nyernek ki valamilyen, a hálózaton a cég központjába küldendő bűvös számokat, hanem igencsak megfogható adatok után kutatnak gépünkön, például a hálózati kártya egyedi száma után.
Eme elrettentő példa ellenére sem tartom rossz ötletnek a Flexlm által nyújtott megoldásokat. Eljöhet az idő, amikor a maihoz képest igen gyors hálózatokon majd különböző programcsomagokhoz férhetünk hozzá, amelyek használatát egy máshol lévő felhasználásiszerződés-szolgáltató fogja engedélyezni, és mi csak annyit fogunk fizetni a programok futtatásáért, amennyi ideig ténylegesen használtuk azokat. Olyan lenne ez, mint manapság a telefonhasználat. A telefonszámlán csak a tényleges beszélgetések ellenértékének összege jelenik meg.
Ehhez természetesen nemcsak gyors hálózatra lenne szükség, hanem kiegyensúlyozott programpiacra, hiszen nem tekinthető normális helyzetnek az, hogy ugyanannak a programnak az ára nullától akár több millió forintig terjedhet. Példaként említhetem az Oracle adatbáziskezelőt, aminek Linuxra írt változatát ingyenesen megkaphattuk otthoni tanulmányozásra, holott piaci ára ennek is a milliót közelíti. Számos nagyon jó program elterjedésének éppen magas ára szab határt. Lehetséges tehát, hogy a Flexlmhez hasonló megoldások a jövőben nemhogy korlátozni fogják a programok terjedését, hanem éppenséggel elősegíthetik, ha segítségükkel leszoríthatjuk az árakat.

Szaló István (ratiosoft@freemail.hu) tanár, immár több mint másfél évtizede foglalkozik programozással, de csak a Java és a Linux megismerése után tudta meg, hogy mi is az igazi programozás. Több írása megjelent már a hazai számítástechnikai lapokban.

Netcraft – és a hajó megy

A Netcraft legutóbbi (2002. januári), webkiszolgálókról készült felmérése szerint változatlanul az Apache-é a működő weboldalak 63,69 százaléka, ami az előző eredményéhez képest 0,35 százalékos növekedést jelent. A Microsoft IIS részesedése 0,55 százalékkal csökkent, jelenleg 26,07 százalékon áll. Mindkettő esetében nőtt a kiszolgálók száma. Az iPlanet a harmadik 2,99 százalék részesedéssel, a negyedik helyen pedig a Zeus áll 2,16 százalékkal – lényegében mindkettő változatlan maradt.
A Netcraft-jelentés a Sun Cobalt leányvállalatát illetően – amely Linux-alapú webkiszolgálókat forgalmaz – jót és rosszat is tartalmazott. „Habár az IP-címek száma a Cobalt-gépeken az elmúlt évben emelkedett, részesedésük a Linuxon futó összes weboldal tekintetében mégis csökkent, jóformán folyamatosan, hónapról hónapra." A Netcraft azt is megjegyezte, hogy Cobaltról két nagy ügyfél is hagyományos Linux gépekre váltott át. A texasi Everyone's Internet internetszolgáltató vállalat nemrég bejelentette „a legnagyobb értékű beszerzési megállapodást, amit független, észak-amerikai internetszolgáltató a Cobalttal valaha is kötött." A vállalat hétszáz Cobalt RaQ-kiszolgálót vásárolt.

Doc Searls

A szabványok szerepe

Néhány szó arról, hogy az újabb szabványok gyakran összeegyeztethetők a szabad és nyílt forrású terjesztéssel.

Szabványok nélkül az Internet Bábel tornya lenne. Az a szabadságjogunk, hogy elmondhatjuk, amit akarunk, alapvetően attól a megegyezéstől függ, hogy ugyanazokat a nyelveket beszéljük. A szabványok adják a közös nyelvi alapot, amelyre a sokszínűség világát építjük. Ahhoz, hogy a programok szemszögéből hasznos lehessen, egy szabványnak világszerte elérhetőnek és mentesnek kell lennie azoktól a koloncoktól, amelyek megakadályozzák széles körű elterjedését.

Vizsgáljuk meg a következményeket egy olyan szellemi tulajdon (például szabadalom vagy szerzői jog) birtokosa szemszögéből, aki ezt a tulajdonát egy ipari szabvány alapjaként szeretné elterjeszteni. Vagy tanulmányozzuk annak a fejlesztőnek az érdekeit, aki megtudja, hogy valaki másnak a szellemi tulajdona akadályozza egy szabvány megvalósításában. Összeférhet-e a szellemi tulajdon birtoklása a szabványokkal a nyílt forráskód világában? Ez nem pusztán elméleti kérdés. A szabványügyi szervezetek szerte a világon arra a kérdésre keresnek választ, hogyan egyeztethető össze a magánkézben lévő szellemi tulajdon a Free Software Foundation (Szabad Szoftver Alapítvány) irányelveivel és a Nyílt Forráskód Meghatározással (Open Source Definition), amelyek szerint a forráskódot nyilvánosságra kell hozni és a programok szabad másolását, módosítását és terjesztését engedélyezni kell.

A szabadalmak jelentik a legnagyobb fenyegetést a szabványok és azok nyílt forráskódú programokban történő felhasználása szempontjából. Bárki, akinek olyan szabadalom van a birtokában, amely egy szabvány megvalósításához nélkülözhetetlen gondolatokra terjed ki, megakadályozhatja, hogy e szabványt megvalósító terméket készítsünk, használjunk vagy árusítsunk.

Nem erőltetem a „megvalósításához nélkülözhetetlen” kifejezés értelmének meghatározását, de gondoljunk bele egy olyan helyzetbe, amikor egy szabvány egyetlen műszakilag kivitelezhető vagy gazdaságilag ésszerű megvalósítása szabadalmazott módszer felhasználását igényli. Mivel a törvény általában nem teszi kötelezővé a szabadalmak átadását, és nem határozza meg az „ésszerű” jogdíj mértékét, a szabvány teljességgel elérhetetlen lehet azok számára, akik nem engedhetik meg maguknak a szabadalom megvásárlását vagy saját fejlesztéssel a technológiai korlát megkerülését.

Sokan adtak hangot tiltakozásuknak a nyilvánosság előtt, amikor a World Wide Web Consortium (W3C) puhatolózásképpen olyan szabadalomelbírálási eljárást javasolt, melynek értelmében webes szabványok részévé válhatnának olyan szabadalmazott módszerek is, amelyekért ésszerű és megkülönböztetést nem jelentő (RAND – Reasonable and NonDiscriminatory) jogdíjat kell fizetni. A szabad és nyílt forráskód közössége arra mutatott rá, hogy ezek a jogdíjak – még akkor is, ha „megkülönböztetést nem jelentő” mértékűek – nem férnek össze a szabad másolást, módosítást és terjesztést megengedő felhasználói engedély mellett a forráskóddal együtt terjesztett programokkal. A közfelháborodás hatására a W3C újrafogalmazza a szabadalmakkal kapcsolatos állásfoglalását. A cikk megjelenésének idején az állásfoglalás új vázlata nyilvános hozzászólások céljából már minden bizonnyal elérhető lesz a ➲ http://www.w3c.org honlapon.

A szabadalmakkal kapcsolatos nehézségek egyik megoldása az lenne, ha a szellemi tulajdon birtokosainak kötelezővé tennék, hogy szabványok megvalósítására a szabadalmakat jogdíj nélkül adják át. A W3C-hez hasonló szervezetek tagjai pontosan ebben állapodtak meg (bizonyos feltételek mellett, amelyeket a honlapjukon olvashatunk). Nem minden szabványügyi szervezet rendelkezik ehhez hasonló elvekkel. A szabványok megvalósításánál mindig ellenőrizni kell a szabványt kibocsátó szervezet szabadalmakkal kapcsolatos felfogását, meggyőződve arról, hogy a szabvány megvalósításának nincs ismert szabadalmi akadálya.

Ha egy szabadalmat egy szabvány megvalósításának céljára adnak át, a szabadalom felhasználási engedélye még akkor sem feltétlenül egyeztethető össze a fejlesztés alatt álló program terjesztési engedélyével. Jellemző például, hogy egyes szabadalmak felhasználási feltételei kimondják, hogy a szabadalom felhasználásának joga csak a szabvány megvalósítására terjed ki („alkalmazási kör”-korlátozás).

A GPL nem fér össze szabadalmak felhasználási engedélyének olyan változataival, amelyek az alkalmazási kör tekintetében megszorításokat tartalmaznak. A GPL-programoknak biztosítaniuk kell azt a szabadságjogot is, hogy a program felhasználásával hozzanak létre újabb programokat, beleértve a más célra használt programokat is (a szakértők úgy fogalmaznak, hogy a kód „elágaztatás” és „újrafelhasználás” céljára is elérhető). A szabadalmak korlátozott felhasználási körű engedélyei a GPL 7. szakaszával ütköznek, amely többek között ezeket mondja ki: „Ha egy szabadalom felhasználási engedélye nem tenné lehetővé a program jogdíj nélküli terjesztését, akkor a szabadalom engedélyének és a jelen engedély tiszteletben tartásának is egyetlen módja az, ha lemondunk a program terjesztéséről.”

Mivel a szabadalmak tulajdonosainak üzleti céljai különbözőek, el kell olvasnunk a szabadalom felhasználási engedélyét és meg kell ismernünk a szabványt támogató szervezet szabadalmakkal kapcsolatos álláspontját. Keressük meg az olyan szabványügyi szervezeteket, mint a W3C és az Embedded Linux Consortium, amelyek kíváncsiak a Nyílt Forráskód Közösségének a szabadalmakkal kapcsolatos elvekhez és eljárásokhoz fűződő véleményére.

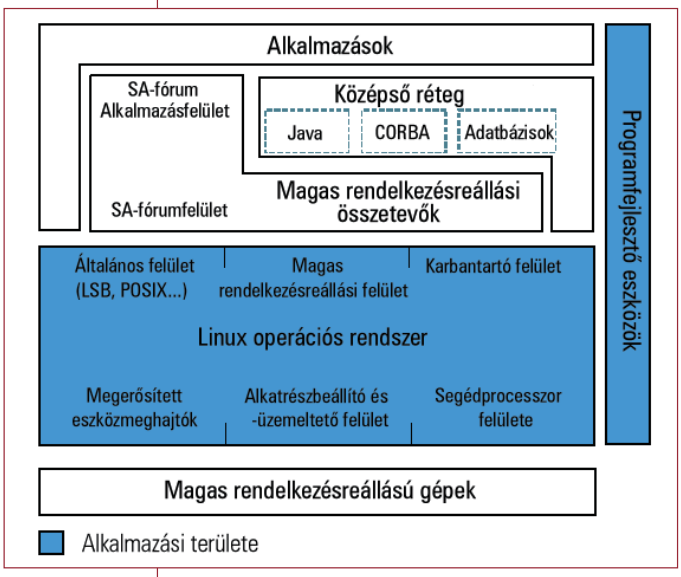
Linux Journal május, 97.szám

Lawrence Rosen (www.rosenlav.com) magángyakorlatot folytató jogász a kaliforniai Redwood Cityben. A Nyílt Forrás Kezdeményezés (Open Source Initiative) ügyvezető igazgatója és jogtanácsosa (➲ http://www.opensource.org).

Beágyazott Linux a távközlési vállalatoknál

Nyújtsd ki a kezed és… érzed, ahogy megérint a Linux? Rick bemutatja, milyen utakon szivárog be a Linux a távközlési piacra.

A 22 tagból álló nonprofit Open Source Development Lab (OSDL) a New York városában tartott LinuxWorld 2002 alkalmával jelentette be, hogy egy új Linux-irányvonallal a távközlési készülékek piacát célozza meg. Az OSDL létrehozta a Carrier Grade Linux munkacsoportot, amelynek „elképzelések és útmutatások" kidolgozása a feladata, valamint az olyan „kereskedelmi és nyílt forrású összetevők fejlesztésének ösztönzése, amelyek az iparág igényeinek megfelelő alkalmazások létrehozását célozzák". A Carrier Grade Linux munkacsoportban



A távközlési ipar elvárásainak is megfelelő Linux felépítése. Az ábrát a megfelelő engedélyek birtokában közöljük

néhány olyan távközlési nagyágyú is képviselteti magát, mint az Alcatel, a Cisco, a HP, az IBM, az Intel vagy a Nokia. Linux-szállítóként a MontaVista Software, a Red Hat és a SuSE is felsorakozik a munkacsoport tagjai közé. Az OSDL számos olyan érvet talált, amellyel egy új, szabványokon alapuló, a távközlési ipar elvárásainak is megfelelő operációs rendszer fejlesztésének szükségessége alátámasztható, valamint hogy ennek alapjaként a Linux jelenti a lehető legjobb választást.

- A hálózatok egyre inkább multimédiás távközlési szolgáltatásokat biztosítanak.
- Nagyobb sávszélességre és új kiépítésekre van szükség.
- A nyílt szabványokon alapuló, készen beszerezhető programösszetevők segítségével az új szolgáltatások gyorsabban piacra dobhatók.
- A nyílt szabványokat támogató szemlélet csökkenti az új kiépítések szerint készülő termékek fejlesztésének költségét és kockázatát.
- A Linux a leggyorsabban terjedő általános célú kiszolgáló operációs rendszer.
- A Linux-rendszermag két részre szakadását mind az adatközpontnak, mind a távközlési piacnak érdekében áll elkerülni.

A csoport elsőként a piaci követelmények felmérését végzi el, majd megtervezi egy távközlési Linux szerkezetét (lásd *ábránkon*), és ösztönzi a különféle, Linuxra épülő összetevők fejlesztését, amelyek eltérő igényekre kínálnak megoldásokat.

Azt, hogy a Linux számára a távközlési ipar ígéretes területet jelent, az alábbi idézetek is fényesen bizonyítják:

- HP: „A Linux a távközlési ipar jövőbeli operációs rendszere." Ugyancsak idén a New York-i LinuxWorldön a HP számos új, az internetszolgáltatók, a távközlési és hálózati szolgáltatók számára fejlesztett Linux-alapú termékkel és szolgáltatással jelent meg. A távközlési kiszolgálók új, Linux-alapú családját is megtaláljuk közöttük, valamint egy fejlesztői készletet a HP Opencall programjához. Az új, kifejezetten a távközlési ipar számára fejlesztett kiszolgálócsalád tagjain – ha elkészül – a Carrier Grade Linux fog futni. *Mark Butler*, a HP távközlési rendszerek részlegének operatív vezetője szerint a HP teljes odaadással támogatja a Linuxot mint távközlési ipar jövőbeli operációs rendszerét. „A Linux a távközlési iparág operációs rendszere lesz a jövőben" – véli Butler. „A HP élen jár a Linux távközlési piacon való elterjesztésében."
- A Motorola meg az új HA Linux alaprendszerrel a hat kilences szintet célozza: a Motorola Computer Group szintén Linux-rendszert fejleszt a távközlési ipar számára. Nemrég jelentették be az iparág számára fejlesztett Linux-rendszerük új változatát, a HA Linux 3.0-t, amely a Motorola szerint „jelentős előrelépést jelent egy olyan operációs rendszer kialakítása felé, amellyel garantálható a hat kilences rendelkezésre állás, vagy ami ezzel egyenértékű, az éves szinten mindössze 30 másodpercet kitevő leállási idő". A Motorola szerint az ilyen rendelkezésreállási szint eléréséhez nemcsak gondosan tervezett gépekre, de megfelelő programokra is szükség van. A Motorola a HA Linuxot mint a távközlési ipar igényeinek megfelelő operációs rendszert kínálja CompactPCI gépeihez.
- A Nokia linuxos rendszereket mutatott be All-IP mobilhálózataihoz: a franciaországi Cannes-ban megrendezett 3GSM World Congress alkalmával a Nokia bejelentette új, Linux-alapú megoldását, amelyet „All-IP" nevű mobilhálózataihoz fejlesztett. Az első All-IP megoldáson alapuló termékek a Nokia nyílt, távközlési cégek számára tervezett FlexiServer és FlexiGateway alaprendszerei lesznek. 2002 januárjában a MontaVista Software bejelentette, hogy a cég közreműködik a Nokia Networks All-IP megoldásának fejlesztésében.

Linus: „kyllä” – jöhet a időosztásos rendszermagfolt!

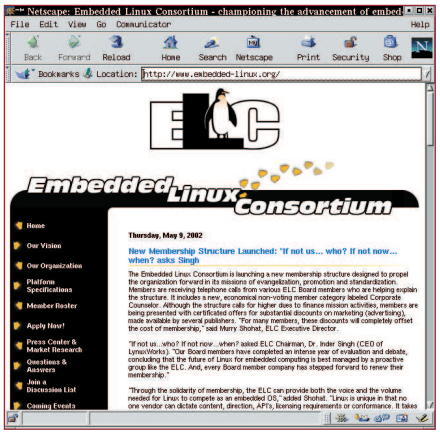
Az eredetileg a MontaVista Software által bemutatott és újabban *Robert Love* által továbbfejlesztett preemptív Linux rendszermagfolt a 2.5.4-pre6 rendszermagváltozattól kezdve hivatalosan is a fő Linux-rendszermag fejlesztési fájának része lett.
Ugyan a fejlesztés eredetileg az ipar és a beágyazott vezérlőalkalmazások igényeinek megfelelően a Linux válaszidejének javítását célozta, ám az eredmények Love szavai szerint nemcsak ezeken a területeken mutatkoznak meg: „…általában véve is jobb rendszert eredményez”. A hagyományosan alacsony válaszidőt igénylő területek mellett – kép- és hangkezelés, beágyazott és valós idejű alkalmazások stb. – egy időosztásos rendszermag előnyei bármely interaktív feladatnál megmutatkoznak. Remélhetőleg hamarosan jobban kezelhető, a felhasználó műveleteire gyorsabban válaszoló asztali rendszerekkel találkozhatunk.

Bangalore: új linuxos zsebtitkár

Egy bangalore-i székhelyű indiai fejlesztőcég jó érzékkel ismerte fel a rést a csúcskategóriás, drága Pocket PC-zsebtitkárok és a kisebb tudású, de elérhető árú Palm-készülékek között, és elkészített egy új, Kaii névre keresztelt PDA-t (a név a „kéz” megfelelője a helyi kannada nyelven). Egy a Bangalore Timesban megjelent írás szerint az Infomart (➲ http://www.infomart.co.in) 200 dollár körüli áron szeretné értékesíteni a 320×240 képpontos, szürkeárnyalatú kijelzővel szerelt modelleket, míg a színes TFT-kijelzővel készülők ára 300 dollár lesz – mindkét típus Linuxot futtat.
Az Infomart mind a felhasználói felület, mind az alkalmazások szempontjából a lehető legmagasabb szintű együttműködési lehetőségre törekszik a Sharp új Zaurus készülékével. Ennek érdekében ugyanazt a programkészletet használták fel: a Lineo Embedix Plus és a Trolltech Qt/Embedded és Qtopia környezetét, az Opera böngészőt, valamint az Insignia Jeode nevű virtuális Java gépét.
A költségek lehető legalacsonyabb szintre szorítása érdekében a Kaii 160 MHz-es Hitachi SH3 processzort kapott, és csak programalapú, vagyis tulajdonosának a kijelzőn megjelenő billentyűzetet kínál.

A Kaii névre hallgató linuxos zsebtitkár

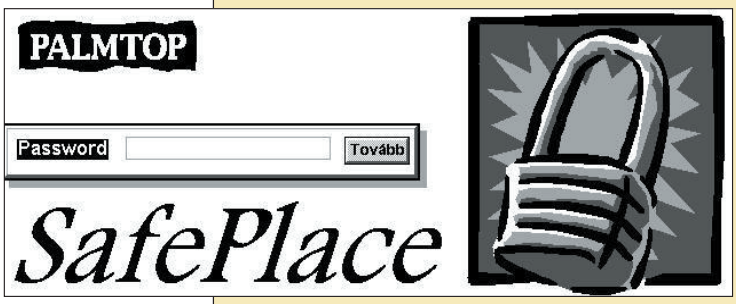
A Kaii emellett változattól függően 32 – 128 MB RAM-mal és 32 MB ROM vagy flashmemóriával gazdálkodhat, USB-vezérlője központi és alárendelt eszközként egyaránt használható, rendelkezik RS232 soros kapuval, IrDA-felülettel és CompactFlash Type II és MMC kártyákhoz használható bővítőaljzatokkal. Az Infomart tervei szerint az USB-csatlakozónak központi szerepet kiosztva a készülékhez majd külső kiegészítők, például billentyűzet vagy céleszközök csatlakoztathatók, így a Kaii nem csak a hagyományos zsebtitkári feladatok ellátására lesz alkalmas. A cég az egész világon keresi gyártó- és forgalmazópartnereit.
Kérdés: lehetséges, hogy a Zaurusszal való együttműködési lehetőség megőrzése új hullámot indított el a linuxos zsebtitkárok piacán?
A linuxos zsebtitkárokkal kapcsolatos legújabb hírek mindig megtalálhatók a LinuxDevices.com „Linux PDAs Quick Reference Guide” útmutatójában, a ➲ http://www.linuxdevices.com/articles/AT8728350077.html címen.

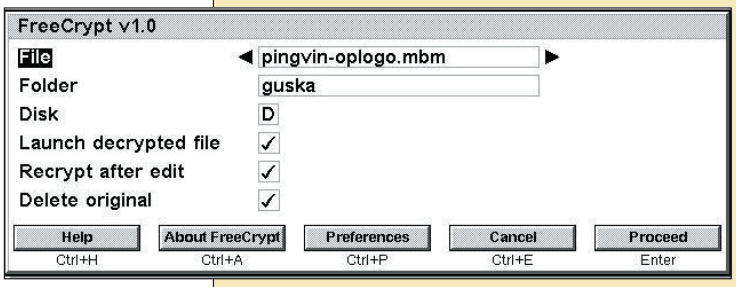


Rick Lehrbaum (rick@linuxdevices.com) hozta létre a LinuxDevices.com „beágyazott Linuxok portálját”, amely nemrég tagja lett a ZDNet Linux Resource Centernek. Rick 1979 óta foglakozik beágyazott rendszerek fejlesztésével. Társalapítója az Ampro Computersnek, alapító tagja a PC/104 Consortiumnak, és fontos szerepet játszott abban, hogy az Embedded Linux Consortium elindulhatott.

Biztonságosan!

Az informatikusok az elmúlt pár év egyedfejlődése során a „kutyus95” típusú jelszavaktól eljutottak a „GCV”!fAZ@,E)J($rW” típusú jelszavakig. A kérdés tehát: milyen is az a megjegyezhető jelszó, amely megfelelően védi a rendszerünket, továbbá hogyan tároljuk és használjuk? A lelkem mélyén lakozó paranoid azt





súgja, hogy a jelszó mindig a 2. példához hasonló bonyolultságú legyen. Ez idáig rendben is van, de ha egy olyan emberre kényszerítjük rá a fenti jelszót, aki még a kutyus95-öt sem képes megjegyezni, csak azt érjük el, hogy felírja egy papírra. Bár a szólásmondás szerint „a szó elszáll, az írás megmarad”, a mi esetünkben legfeljebb így hangozhatna: „a szó elfelejtődik, a papírra felírt jelszó viszont idegen kézbe kerülhet”. Az eszményihez közelítő állapot, ha bonyolult, általunk sem megjegyezhető jelszót választunk, amelyet egy PDA-n vagy egy titkosított állományban helyezünk el. Bár mindez bonyolultnak és fölöslegesnek tűnhet, ám ha belátjuk, hogy minden géphez külön jelszót kell használnunk, vagy akár minden karbantartáshoz is másikat, a fenti eljárás mindenképpen indokolttá válik. Ha jelszavainkat PDA-n tároljuk, nagyon fontos, hogy olyan alkalmazást válasszunk, amely az adatbázist is megfelelően védi. Számtalan olyan alkalmazás létezik, amely nemcsak az adatbázist (a jelszavainkat) rejti kódoltan, hanem a memóriában is így tárolja őket; és ha az alkalmazás kikerül a működés középpontjából, azonnal jelszóval védi. Ebben az esetben a PDA-t magát érdemes védenünk a lopás és a hozzáférés ellen, viszont ha mégis elhagynánk, akkor sincs nagy baj, csak mielőbb meg kell változtatnunk hozzáféréseink jelszavait. A PDA azonban egy életstílus, amely igen költséges és sokan fölöslegesnek tartják. Ezzel szemben a kereskedelmi forgalomban egyre elterjedtebbek lettek a különféle memóriakártyás megoldások. A sok-sok CF-alapú kártya közül mindenki kiválaszthatja magának a legkedvesebbet, amely méretével illeszkedik az életstílusához, valamint a gépével is együttműködik. Miért is jó kivehető (nem felejtő) memórián tárolni a jelszavakat? Mert bármikor bárhol kéznél van, mert titkosított állományrendszert rakhatunk rá – és mert olcsó. Számomra az USB-felületre csatolható PEN DRIVE volt a legszimpatikusabb, amely 15 000–20 000 forintos árával még az elfogadható kategóriát képezi, és USB-felület szinte mindegyik gépben megtalálható. Ez egy olyan tollalakú eszköz, amely nem nagyobb egy kihúzófilcnél – 64 MB és 128 MB összeállításban kapható. Az USB-kapura csatlakoztatva az `usb-storage` modullal tudjuk meghajtani, amely *0*-ként csatolható rendszerünkhöz. Innentől kezdve adva van a lehetőség, hogy titkosító fájlrendszert rakjunk rá, ezzel is biztosítva magunkat, ha esetleg illetéktelen kezekbe kerülne. A két hónappal ezelőtti számban egy igen részletes írás szólt arról, hogyan készítsünk ilyen rendszert, én mégis a loop-aes rendszer megépítését ajánlom, mert GPL-es program esetén ezen biztosított, hogy titkosító rendszerünk minden rendszermagváltozat alatt használható, illetve olvasható lesz. Telepítése egyértelmű, a letöltés után leírását a README állományban megtaláljuk. A titkosító fájlrendszer elkészítése után még ne dőljünk hátra, hiszen nem tettünk meg mindent a biztonságért. Ha elhagyjuk vagy ellopják, a CFS (titkosított fájlrendszer) véd minket, de ha ezt a GPG biztonságával is kombináljuk, talán végre nyugodtan alhatunk. Így minden jelszót tartalmazó állomány eleve egy titkosított rendszeren tárolódik, ráadásul GPG titkosított állományként. Kínálja magát a lehetőség, hogy SSH-kulcsainkat is ezen a médián tároljuk, így ha kiugrunk ebédelni, a gépből csak kihúzzuk a kártyát, és a nyakunkba akasztva biztosak lehetünk benne, hogy aki ebéd közben a gépünk elé ül, az ma kulcsok nélkül fog távozni.

Éljen a paranoia!

Kapcsolódó címek

- ➲ http://loop-aes.sourceforge.net/
- ➲ http://www.pendrive.com
- ➲ http://www.gnupg.org
- ➲ http://www.openssh.org
- ➲ http://www.crypto.com

Varga S. Csaba (guska@guska.hu) az 1.1-es Slackware óta linuxozik. Kedvtelései közé tartozik a fotózás és Linux telepítése PDA-kra. Legszívesebben a Gerecsében túrázik a barátaival.

Ő mondta

Ha negyedóráig mindenki híres lenne a való világban, a Világhálón bárki ismertté válna 15 ember számára. *(David Weinberger)*

Főzzünk finomakat

Ypsilanti, Michigan, 2002. február 4. – a nyílt forrású Linux operációs rendszer lett egy népszerű, új receptszolgáltatás, a ➲ http://recipesbyemail.com szíve.
A Tap Internet által fejlesztett szolgáltatás révén egy levelezőügyféllel bárki könnyen és gyorsan receptek százaiban végezhet keresést.
„Eredetileg egy ügyfelünk számára készítettünk elképzeléseink egyikének szemléltetésére mint elektronikus levél útján kereshető adatbázist" – mondta *Michael Kimsal*, a Tap Internet igazgatója.
Az alaptechnológia elkészülte után azonban hamarosan úgy döntöttünk, hogy mintapéldányt állítunk fel, amelyet más ügyfeleknek is megmutathatunk. Az internetfelhasználók hamarosan rákaptak arra, hogy a ➲ http://recipesbyemail.com segítségével találják meg a keresett recepteket.
A rendszeres felhasználók jelentős százaléka vak, amelynek elsősorban az a magyarázata, hogy számos weboldalt úgy elborítottak a látványos JavaScript- és Flash-animációk, hogy a szöveg-beszéd-átalakító rendszerekre utalt felhasználók képtelenek őket használni. „Nagyon meglepődtem, amikor levelet kaptam ezektől a felhasználóktól, mivel erre korábban nem is gondoltam" – mondta Kimsal. „Szándékosan csak szöveges leveleket fogadunk el, hogy a lehető legjobb együttműködést biztosítsuk, függetlenül attól, hogy valaki Outlook, Palm Pilot vagy mobiltelefon segítségével ér-e el minket. Annak köszönhetően, hogy a Linuxra alapoztuk, mindezt egy puszta dróttal valósíthattuk meg, és mégis hasznos dolgot nyújtunk a felhasználóknak, akik közül sokan nem rendelkeznek gyors kapcsolattal vagy a legújabb böngészőkkel."
A rendszer Slackware Linuxra épül Perl, MySQL, Procmail és PHP felhasználásával. „Hamarosan új lehetőséggel bővül a szolgáltatás, és a felhasználók saját receptjeiket is beküldhetik, ami reményeink szerint tovább növeli majd a rendszer iránti érdeklődést."
➲ http://recipesbyemail.com

Linux-index

1. Ennyi milliárd dollár értéket tesznek ki a Pentagon könyveletlen tranzakciói: 2,3
2. Hozzávetőlegesen ennyi millió dollárt pazarolt el a Pentagon az elmúlt két percben: 2
3. A Plútó bolygónál nagyobb átmérőjű holdak száma a naprendszerben: 8
4. A Föld holdjának átmérője ennyi kilométerrel nagyobb a Plútóénál: 1,176
5. Ennyi óra múlva érkezik meg az első levélszemét (spam) azt követően, hogy egy levélcímgyűjtő-alkalmazás rátalált egy frissen kitett weboldalra: 8
6. Becslések szerint a globális felmelegedés a Föld forgását 2100-ra ennyi tízezred másodperccel fogja lassítani: 1
7. Ennyi százalékkal növekszik a világ 65 év feletti népessége 2025-re: 100
8. Ennyi százalékkal emelkedik a gyermekek száma a világon 2025-re: 3
9. Ennyi nyilvános, vezeték nélküli világháló-elérési pont felállítására kerül sor Koreában a nyári labdarúgó világbajnokság kezdetéig: 25 000
10. A felhasználók ennyi százaléka éri el a weboldalakat közvetlen megadás vagy könyvjelzők útján (keresőmotorok használata helyett) a 2002. február 6-i adatok alapján: 52
11. Ugyanez egy évvel korábban: 46
12. Ennyi millió dolláros nyereséget jelentett be a Yahoo 2000-ben: 71
13. Ennyi milliárd dolláros veszteség érte volna a Yahoot, ha a lehetőségek költségét beszámították volna: 1,3
14. A mérnökök ennyi százalékánál vált bizonyossá a 10. osztály előtt vagy alatt az, hogy milyen pályát választanak: 59
15. A mérnökök ennyi százaléka „választotta ezt a pályát azért, mert vonzódott a matematikához és a természettudományokhoz, és a mindennapi tevékenységeket új megoldásokkal szerette volna gazdagítani ": 79
16. Ennyi milliárd dollár forog aprópénzben az Egyesült Államokban: 7,7
17. A Walmart.com ennyi operációs rendszert kínál 399 amerikai dollár értékű PC árukapcsolásával: 0
18. A Wal-Mart üzletek száma világszerte: 4382
19. A Linux-felhasználók becsült száma legkevesebb fő: 2 403 060
20. A Linux-felhasználók becsült száma legfeljebb fő: 60 076 500

Források

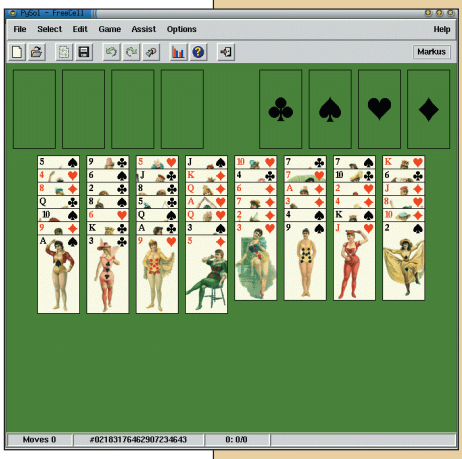
1–2.: CBS News, 2002. január 29.
3–4.: Solarviews.com
5–6.: DSL Reports (➲ http://www.dslreports.com)
6.: Astronomy.com
7–8.: United States Census Bureau (az USA népszámlálási hivatala)
9.: Wireless World Forum
10–11.: ZDNet UK, a WebSideStory adatai alapján
12–13.: Fortune
14–15.: MathSoft-felmérés, 1200 mérnök megkérdezése alapján ➲ http://www.mathcad.com
16.: Business 2.0
17–18.: Wal-Mart
19–20.: Linux Counter, 2001. február 21.

Éljen a nyár?

Itt a nyár – ki az, aki ilyenkor munkával szeretne foglalkozni? Én nem, azonban akadnak olyan elvetemültek (nem is kis számban), akik ilyenkor sem pihenhetnek – akár jó szántukból, akár szándékaik ellenére. Én az utóbbiak közé tartozom. Mi ilyenkor a követendő példa? Ezt már őseink is kitalálták, évtizedek óta bevett gyakorlat. Tegyünk úgy, mintha dolgoznánk, de valójában múlassuk a nyáridőt. Nem is való másra. Nézzük át, mivel is lehetne az időt a legjobban elütni!

1. számú túlélési szabály: fő a nyugalom!

Márpedig mi lehetne jobb az áldott nyugalmi állapot eléréséhez, mint egy kis pasziánsz? Természetesen csupán a teljes ellazulás elérésének érdekében, Linux alatt. Nos, erre egy jó kis programot tudok ajánlani: a pySOL-t. (➲ http://www.oberhumer.com/opensource/pysol/)



Csak szerényen: szerintem ez a pasziánsz programok királya. Több mint kétszáz játékot ismer, továbbá bővíthető a felhasználók által írt játéktípusokkal is. Akik még ennél is többre vágynak, azok számára elérhető a beépített HTML-alapú segítség és a háttérzene is. Bár én inkább maradok az MP3-nál. Természetesen a program zenét szolgáltató részével az MP3 is elérhető, de én jobban kedvelem az XMMS-t. Tapasztalataim szerint az öröm sajnos sosem felhőtlen. Majd' kitéptem a maradék hajamat, amikor az egyik barátom édesanyja – otthon használatos Linuxukra telepítettem a programot – közölte velem, hogy jó-jó, szép a program, de már pasziánszmérgezése van, és inkább Mahjonggot szeretne.

2. számú túlélési szabály: kotord elő a kínai játékokat!

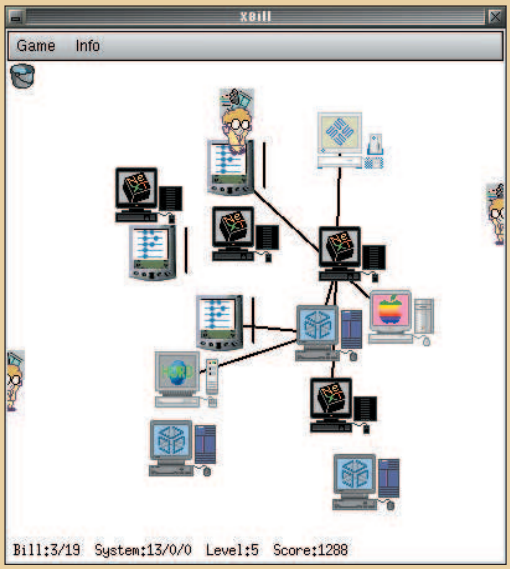
Nos, mit tehet az ember a fenti esetben? Természetesen utánanéz, hogy lehet-e segíteni szegény megfáradt asszonyon, aki az állandó pasziánszpartiktól kiütést kapott. Eszébe jut a KDE, illetve a Gnome alá fejlesztett Mahjongg játék. Rögtön utána bevillan a „linuxos intelmek" közül az első: ahova nem muszáj, oda ne tegyünk fel KDE-t és Gnome-t! Ennek igazolására két ok létezik: a gyenge gépeken az erőforrások teljesítménye olyan mértékben romlik, hogy az embernek sírni támad kedve. Másodszor folyamatosan tanulj, keresd a választási lehetőségeket. A megszokás az egyik legrosszabb dolog, ami az emberrel történhet. Eme rövid filozófiai értekezés után keressünk rögtön más lehetőségeket. Az alternatívák nem mindig esnek messze a már megszokott dolgoktól, vagyis minden (kártya és táblás) játék a PySOL-hoz vezet. Ennek bizonyítéka a ➲ http://www.telestream.com/~grania/pysol/flowers.html címen érhető el. Mire is bukkanhatunk? 102 fajta Mahjongg-ábrázolásra, tarokkjátékra, egy idáig biztosan nem túlságosan ismert indiai játékra, és a gyermekkorunkból előbukkanó tili-tolijátékra, amelyben négyzetekre szabdalt és a darabokat összekeverten ábrázoló képeket kell a helyes sorrendben kiraknunk.

A Flowersol a PySOL módosított változata, mely a fenti játékokat is ismeri, a Flowersol alatt található játékok nagy része pedig megegyezik a PySOL alatt szereplőkkel. Ha ezt a programot is feltettük, nincs más dolgunk, mint a következő nehézségre várni és tovább játszani.

3. számú túlélési szabály: játsszatok többen!

Na, igen. A barát mamája visszajön, hogy jó ez a Mahjongg meg a tarokk és a többi, csakhogy jó lenne az is, ha a barátnőivel is tudna játszani az Interneten keresztül… Ekkor két eset lehetséges. Az egyik tibeti kolostorok végiglátogatása lelki nyugalmunk visszanyeréséhez, a másik, hogy ismét körülnézünk a Világhálón, hogy milyen lehetőségeket tartogat még számunkra. Nos, ilyet is találunk Linux alá. Ezek közül a legjobban kivitelezett a Mah-Jong, mely a ➲ http://www.stevens-bradfield.com/MahJong/ címen érhető el. Ez a számunkra oly fontos hálózati lehetőségekkel is rendelkezik. A program alkotója létrehozott egy Mah-Jong-kiszolgálót. Ha elindítunk egy játszmát, a többiek csatlakozni tudnak hozzánk, ugyanúgy, ahogy mi is tudunk csatlakozni a többiek által elindított programhoz. Természetesen állandó IP-cím szükséges hozzá, hogy a gép hálózati címéről ne kelljen mindig egyeztetni. Ennek ugyanis valószínűleg nem minden felhasználó képes utánanézni. Belső hálózaton viszont kitűnően lehet vele múlatni az időt. Ne tévesszük szem elől a tényt, hogy ez a Mah-Jong nem az a Mahjongg játék, hanem *a valódi*. Eltér

tehát a megszokottól, mert mindenképpen négy játékos szükséges hozzá, és nem elég a hasábokat párosítani. Éppen ebben rejlik a vonzereje: olyan új játékot tanulhatunk meg, melyet akár társaságban is tudunk játszani, valódi tárgyakkal.

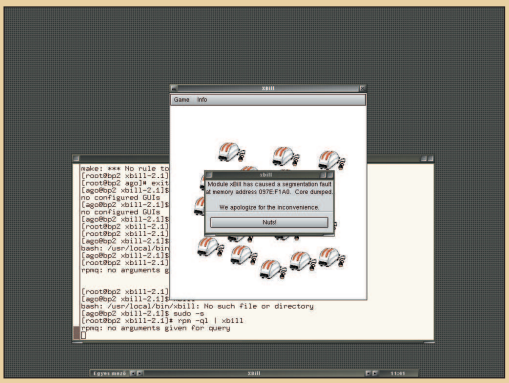


4. számú túlélési szabály: semmit se vegyünk komolyan

Ha a főnök ráunt arra, hogy mi csak pasziánszozunk, vagy a világ másik végén lévő barátunkkal mahjonggozunk, és hangosan követeli, hogy töröljük le a programokat a gépről, mert többé nem akarja látni, hogy ezekkel játszunk, nyugodtan mondjunk igent. Majd megnyugszik, és minek fölöslegesen idegesíteni? Álljuk a szavunkat és töröljük a programokat, majd ugyanezzel a lendülettel tegyük fel a már gyermekkorunkból ismert torpedót (battleships). Ha körülnézünk a Világhálón, ezt a szót beírva egy olyan programot találhatunk, mely a következő igénylistát elégíti ki: egyedül és többen is lehet játszani – az utóbbit hálózaton keresztül is megtehetjük, ráadásul jól néz ki. Nos, a Battala Naval programot erre találták ki (➲ http://www.batnav.sourceforge.net/batnav-en.html). A program egyszerűen csodálatos, bár akad valamennyi hátránya neki is: meg kell szegnem az előbbi fogadalmamat és a Gnome egyes részeit fel kell telepítenem. Nekem ez nem jelent gondot, ahogy a legtöbbünknek sem, de előfordulhatnak olyanok is, akiknek esetleg még 200-as gépük van. Mindenesetre nem olyan borzasztó a program erőforrásigénye, és a legszebb, hogy Debian-csomaggal is rendelkezik. Nem is pazarlok több szót rá, inkább mindenki telepítse és játsszon vele!

5. számú túlélési szabály: éld ki a felgyülemlett erőszakot!

Ha a főnököd a torpedózás örömeitől is megfosztott, jogos a felháborodásod. Hát már szórakozni sem lehet nyáron? Ahelyett azonban, hogy hátrányos helyzetünk miatti elkeseredésünkben fizikai erőfölényünket éreztetnénk vele, vegyünk elő egy réges-régi, ma már a legendák ködébe vesző programot, mely a xbill névre hallgat. Röviden: ez a végső megoldás. Már hat éve is megvan, hogy játszottam vele – a mostani állapot visszatérés a gyökerekhez. Azóta nagyot haladt előre a világ.



Átírták Java Applet formátumra és iPaqon is (természetesen Linux alatt) futtatható. A játék története a következő: gonosz Bill, akit egy kis szemüveges emberke alakít, megpróbálja megfertőzni a számítógépedet egy vírussal, amit operációs rendszernek álcáz. A számítógépeken lévő rendszereket megpróbálja a saját kis vírusára lecserélni és elvinni a régi operációs rendszert a gépről, nehogy újra tudd telepíteni. Sajnos klónozással szaporodhat, mert rengeteg van belőle. A magasabb szinteken több és több jön belőle, egyre gyorsabban. Külön nehézség, hogy a vírus nagyon leleményes. Ha a számítógép, amelyet megfertőzött, hálózatba van kötve, azon keresztül is megpróbál terjedni, egyre több gépet megfertőzni. Amit mi tehetünk, annyi, hogy klónjait szolid kis pofonokkal megpróbáljuk visszatartani a tettétől. Ha ez sikerül, máris magasabb és magasabb szintekre léphetünk. Én személy szerint a kilencedik szintnél nem jutottam tovább, de hat év távlatából nem esküdnék meg rá, hogy sohasem volt példa az ellenkezőjére. Ami még megnyugtató a programban, hogy főnökünk, miután meglátja, hogy mindenki körénk gyűlt és nem dolgozik, valószínűleg elküld egy hét szabadságra, amíg elül az xbill-láz (feltételezvén egyelőre más még úgysem ért a Linuxhoz). Azonban gondoljunk kollégáinkra is: telepítsünk számukra Linuxot, hogy a fentebb felsorolt programokat élvezni tudják. Majd valóban menjünk el szabadságra, mielőtt a főnök rájön, mit is cselekedtünk. Figyelmeztetés: a fent felsorolt csínyekből eredő munkahelyvesztésekért a szerző felelősséget nem vállal!

Deim Ágoston (ago@lsc.hu)
Kedveli a sört, szereti a futást és imádja Szabó Lőrinc verseit. Nem hisz vakon egyik rendszerben sem. Vonzódik a BSD-hez is. Tagja az LME-nek és a MBE-nek. Mottója: a gép nem lehet fontosabb az embernél.

Új termékek

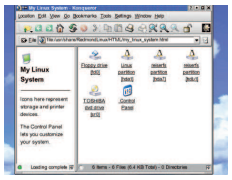
CrossOver Plugin v1.1

A CodeWeavers Inc. CrossOver Pluginjának 1.1-es változata egy olyan átalakító, amely windowsos böngészőbővítményeket és levelezőprogramokat tesz használhatóvá Linux alatt. A program segítségével MS Office- és eFax-fájlokat nyithatunk meg tetszőleges KDE- és Gnome-alkalmazásból, ráadásul az 1.1-es változat már a Windows Media Player folyamait is támogatja. Az 1.1-ben más újdonságok is megjelentek, például minden TrueType-betűkészletet elfogad, kezeli a RealPlayer-fájlokat, a Trillian-bővítményt, a Yahoo Messengert, az iPIX-bővítményt és a vegyiparban használatos Chime-bővítményt. A CrossOver program támogatja a legelterjedtebb böngészőket, többek közt a következőket: Netscape, Mozilla, Konqueror, Opera, SkipStone és Galeon.

e-mail: sales@codeweavers.com,
➲ http://www.codeweavers.com

Desktop/LX

Megjelent a Lycoris (korábban Redmond Linux Personal) új Linux-terjesztése, a Desktop/LX, amely két változatban kapható. A Standard kiadás tartalmazza a Desktop/LX CD-t, a 30 oldalas telepítési kézikönyvet és 60 napos elektronikus leveles támogatást.



A Deluxe kiadásban az előbb felsoroltakon kívül a forráskódokat tartalmazó korong és a DevTools CD is benne van, ennek segítségével a Desktop/LX Deluxe fejlesztőkörnyezetként is használható. A grafikus telepítőprogram magától felismeri a támogatott videó-, hang- és hálózati eszközöket, valamint a csatlakoztatott nyomtatókat. A Desktop/LX beállítása könnyen és gyorsan elvégezhető, és olyan varázslót tartalmaz, amely az internethozzáféréssel rendelkező felhasználók számára a legújabb programváltozatok és frissítések letöltését teszi lehetővé.
e-mail: sales@lycoris.com
➲ http://www.lycoris.com

PortServer CM

Megjelent a PortServer CM, a Digi International új termékcsaládja, amely az adatközpontok karbantartásához nyújt segítséget. Az állványba szerelhető 1U méretű 32-kapus PortServer CM a rendszergazda számára lehetővé teszi, hogy a hálózatra csatlakozó eszközöket a vállalat hálózatának bármely pontjáról figyelemmel kísérje és vezérelje. A kapcsolat fajtája többek közt szabványos TCP/IP a helyi ethernetes hálózaton, vagy telefonvonalon keresztüli modemes kapcsolat lehet. A biztonságról az SSH 2-es változata gondoskodik. A következő protokollok támogatottak még: DHCP, PPP, SLIP, NTP és FTP. Az eszköz memóriája 64 MB SDRAM-ból és 4 MB flashmemóriából áll (bővíthető).

➲ http://www.digi.com

VT100 Set-Top doboz

A VT Media Technologies megjelentette VT100 Edge elnevezésű set-top dobozát. Az Edge sokféle adatfolyamot (például ethernet 10/100) képes kompozit analóg videó vagy digitális S-Video jelekké alakítani, amelyek minden szabványos tévékészüléken megjeleníthetők. A doboz teljes értékű böngészőprogramot tartalmaz, támogatja az összes bővítményt, valamint katódsugárcsöves monitorokkal is képes együttműködni. Az Edge-hez kiegészítőként DVD-lejátszó, CD-újraíró, hajlékonylemezes meghajtó és szabványos IDE-eszköz választható. Támogatja a National and Century Embedded Software által bejelentett Web-Media programot. Az Edge-hez való fejlesztői készlet a VT honlapjáról tölthető le.

e-mail: sales@vtmt.com,
➲ http://www.vtmt.com

PS-R1242, Dual Xeon 1U kiszolgáló

Az Intel E7500-as lapkakészletén alapuló PS-R1242 kiszolgáló két Pentium Xeon processzort tartalmaz, melyek legfeljebb 2,2 GHz sebességűek lehetnek. A kiszolgáló teljesítménye a szuperszámítógépekével vetekszik, ezért elsősorban a szórakoztatóipar (média) és a tudományos számítások területén érdemes használni. A PS-R1242 választható SCSI Ultra160 vagy ATA 100 RAID lemezvezérlővel, valamint két IDE-, vagy három SCA menet közben cserélhető lemezegység-bővítőhely

állítható be rajta. A kiszolgálóba legfeljebb 8 GB kétirányú (two-way interleaved) DDR SDRAM szerelhető. További jellemzők: 1 GB ethernetkapu, egy Fast ethernetkapu az alaplapra építve, két szabad PCI 133/100 bővítőhely és 350/400 W leállítás után cserélhető tápegység.
e-mail: sales@promicrosystems.com,
➲ http://www.promicrosystems.com

NASAS-2040

Az Ateonix Networks bemutatta NASAS-2040 nevű hálózati tároló-eszközét, amelyet kis- és közepes méretű vállalatok igényeihez terveztek. Az eszköz legfeljebb négy – bármilyen gyártótól származó – ATA/100/133 merevlemezt támogat tetszőleges elrendezésben.
Az egyes merevlemezek tárterülete 128 petabájt lehet, és nem szükséges azonos méretű merevlemezeket használni. A bővítéseket a vállalat saját alkalmazottai a helyszínen is elvégezhetik. A merevlemezek az előlapon keresztül ki- és beszerelhetők. Az összes programfrissítés – beleértve a hálózati operációs rendszert, az eszközmeghajtókat és a felhasználók beállításait – webes felületen keresztül tölthető le.
A NASAS-2040 minden programja flashmemóriában helyezkedik el. Az eszköz támogatja a RAID 0, 1 és 5 elrendezéseket.
e-mail: sales@ateonix.com
➲ http://www.ateonix.com

Linux Journal május, 97. szám

A hónap szakmai tanácsai

Hiszem, hogy van /dev/st0!

Nem tudom elérni a szalagos meghajtót! A `dmesg` az alábbi eredményt adja:

```
scsi0 : Adaptec AHA274x/284x/294x
        (EISA/VLB/PCI-Fast SCSI)
        ➥5.1.33/3.2.4
       <Adaptec AHA-294X Ultra SCSI
        ➥host adapter>
scsi : 1 host.
  Vendor: ARCHIVE  Model: Python
  ➥28454-XXX  Rev: 4ASB
  Type:   Sequential-Access    ANSI
  ➥SCSI revision: 02
  Vendor: FUJITSU   Model: M1606S-512
  ➥Rev: 6236
  Type:   Direct-Access        ANSI
  ➥SCSI revision: 02
Detected scsi disk sda at scsi0,
➥channel 0, id 3, lun 0
scsi : detected 2 SCSI generics 1
➥SCSI disk total.
(scsi0:0:3:0) Synchronous at 10.0
➥Mbyte/sec, offset 15.
SCSI device sda: hdwr sector= 512
➥bytes.
Sectors= 2131992 [1041 MB] [1.0GB]
```

A SCSI-merevlemez elérhető, de a szalagos meghajtó nem. Az `eth0` és az `aic7xxx` egyaránt a 9-es megszakítást használja. A SCSI-szalagmeghajtók támogatása be van fordítva a rendszermagba. A gépem egy Pentium 133-as.
Andy Prowse, azp80@amdahl.com

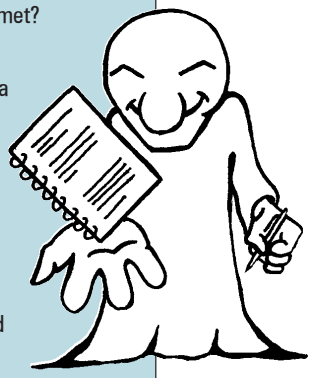
A `dmesg(8)` kimenetének általában tartalmaznia kell egy `st0`-ra vonatkozó sort, közvetlenül az `sda`-bejegyzés alatt. Az én rendszeremen például ez a következő:

```
Detected scsi tape st0 at scsi0,
➥channel 0, id 6, lun 0
```

Szalagos meghajtódat a rendszermag általános (`generic`) eszközként ismerte fel, azonban az `mt(1)` használatához aktív szalag meghajtó program is kell. Ellenőrizd a rendszermag fordítási beállításait! A rendszermag változatától függően a `SCSI support` vagy a `SCSI tape support` alatt érdemes körülnézned.
Chad Robinson, crobinson@rfgonline.com

Jogosultságváltozás a /etc könyvtáron

A */etc* könyvtáram jogosultsága véletlen időközönként 600-ra változik. Néha egész nap minden rendben van, máskor pár perccel azután, hogy a */etc* jogosultságát 755-re állítom, visszaváltozik 600-ra. Eleinte azt hittem, hogy ez a rendszermag védekezése hibás memória vagy lemezegység ellen. Kicseréltem a memóriát, csak egy az alaplapgyártó által minősített modult hagytam az alaplapon, azonban próbálkozásom eredménytelen maradt. A merevlemezt nem áll módomban kicserélni. A futó folyamatok számát a mellékelt fájlban felsoroltakra csökkentettem. Normális-e ez a viselkedés, okozhatja-e a meghibásodott számítógép vagy rosszul telepített alkalmazás, esetleg feltörték a rendszeremet?
Stan Katz, stan@cloud9.net



A számítógép meghibásodása szinte soha nem okoz ilyen jellegű gondot, és semmiképpen nem ilyen ismétlődő módon. Ideje megkeresni azt a vélhetően rosszindulatú alkalmazást, amelyik a hiba oka lehet. Vizsgáld meg a többi naplófájlt is, talán valamilyen nyomra bukkansz bennük. Csak a rendszerindítás üzeneteit küldted el, ez azonban semmit sem mond a rendszer működéséről. Nézd végig a `crontab` fájljait, kövesd nyomon, hogy mely alkalmazások indulnak innen. A rendszerdémonok és a gyakran használt programok bináris fájljait hasonlítsd össze a telepítőlemezen lévő változatukkal, nem lettek-e trójai programokkal lecserélve. Végül keresd meg a `setuid root` beállítású alkalmazásokat – ezek gyakran arra utalnak, hogy rendszeredre betörtek. Ámbár elég furcsa viselkedés egy trójai program részéről, hogy könyvtárak hozzáférési jogosultságait korlátozza. Mivel rendszeredhez és a telepített fájlokhoz, illetve a naplófájlokhoz nem férünk hozzá, nem tudunk pontosabb megoldással szolgálni.
Chad Robinson, crobinson@rfgonline.com

Szinte biztosan egy cronfeladat állítja át a jogosultságokat, vagy ami még rosszabb, egy rosszindulatú rendszermagmodul garázdálkodik a gépeden. Tedd megváltoztathatatlanná */etc* jogosultságait a `chmod 755 /etc; chattr -i /etc` paranccsal, ami remélhetőleg a hibát okozó program sikertelen futását fogja eredményezni (cronfeladat esetén még levelet is kapsz a hibáról). Esetleg próbálkozz az `rgrep -r chmod /etc /var/spool/cron` paranccsal, ennek segítségével is megtudhatod, hogy mi változtatja meg a jogosultságokat.
Marc Merlin, marc_bts@valinux.com

X-et szeretnék használni, de nincsenek érvényes beállításaim

Nem tudom használni az X-et Toshiba Satellite Pro 4600 gépemen, amelyben Trident CyberBladeXP videokártya van. Red Hat 7.2-t használok, és a következő hibaüzenetet kapom:

```
Fatal server error:
No Valid modes found.
```

Troy, coder@starmail.com

A Red Hat weboldalán megjelent egy frissítés, amelyben említést tesznek a videokártyával kapcsolatos hiba megoldásáról. Próbáld meg frissíteni a szükséges csomagokat, és ezután próbálkozz újra. További adatok a következő címen olvashatók:
➲ rpmfind.net/linux/RPM/redhat/updates/7.2/i386/Xconfigurator-4.9.39-2.i386.html
Mario Bittencourt Neto, mneto@buriti.com.br

Linux Journal május, 97. szám

A Linux Journal honlapján számtalan gond megoldásához találhattok további segítséget. A Sunsite tüköroldalait, a gyakran feltett 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.

Az időosztásos rendszermag bemutatása

Ha csak magasabb pontszámot szeretnénk elérni a Quake-ben, netán audiorajongók vagyunk, esetleg simábban futó felületet szeretnénk – a rendszermag lappangási idejének csökkentése igen fontos cél.

A teljesítmény mérése két szempont: az áteresztőképesség (throughput) és a lappangás (latency) alapján történik. Az első olyan, mint az autópálya szélessége: minél szélesebb, annál több autó utazhat rajta. A második pedig a sebességkorláthoz hasonló: minél magasabb, annál hamarabb érnek az autók A pontból B-be. Feladatok esetében nyilvánvalóan mindkét érték fontos. Néhány munka azonban olyan lehet, hogy az egyik érték szükségesebbé válik a másiknál. Az autópályás hasonlatnál maradva a teherfuvarozás érzékenyebb lehet az áteresztőképességre, ugyanakkor a futárszolgálat számára a lappangás a fontosabb. E cikk fő témája pontosan ez a jó kis régivágású rendszermagfeladat: a lappangás csökkentése és a rendszer válaszidejének növelése.

Az audio-, illetve videofeldolgozás és -visszajátszás azon feladatok közé tartozik, amelyek sokat nyerhetnek az alacsonyabb lappangási idővel. A Linux számára egyre fontosabb, hogy az interaktivitás teljesítménye is növekedhet. Nagy lappangási idő esetén az egérkattintások és a hasonló felhasználói tevékenységek meglehetősen sokáig válasz nélkül maradnak, ami igen távol esik attól a pattogósságtól, amit a felhasználók elvárnak. A rendszer ugyanis a fontos folyamatokat nem tudja elég gyorsan elérni.

A gond – legalábbis a rendszermag vonatkozásában – az, hogy nem időosztásos módon működik. Általában ha valami fontos történik, például bekövetkezik egy interaktív esemény, a fogadóalkalmazás fontosságnövelést kap, következésképpen futni fog. Így működnek az időosztásos módon többfeladatosított (multitask) operációs rendszerek. Az alkalmazás addig fut, amíg egy adott időmennyiséget fel nem használ (ezt nevezik időszeletnek), vagy valamilyen más fontos esemény nem történik. A másik lehetőség az együttműködő többfeladatos módszer (cooperative multitasking), ahol – mielőtt egy új folyamat elkezdhetne futni – az alkalmazásoknak maguknak kell jelenteniük, hogy „kész vagyok!”. A gond az, hogyha valami a rendszermagban fut, az ütemezés gyakorlatilag ilyen együttműködő típusú lesz.

Az alkalmazások, akár a felhasználótérben futva hajtják végre saját kódjukat, akár a rendszermagban hajtanak végre rendszerhívásokat, vagy más módon dolgoztatják a rendszermagot, e két módszer valamelyike szerint működnek. Amikor egy folyamat a rendszermagban dolgozik, addig fut, amíg úgy nem dönt, hogy megáll – eközben az időszeleteket és a fontos eseményeket figyelmen kívül hagyja. Hiába válik egy még fontosabb folyamat futtathatóvá, akkor sem tud lefutni addig, ameddig a jelenleg futó folyamat ki nem lép, már ha épp a rendszermagban tartózkodik. Márpedig ez a folyamat milliszekundumok százait fogyaszthatja.

Lappangási megoldások

Az első és legegyszerűbb megoldás a lappangás problémájára, ha újraírjuk a rendszermag algoritmusait, hogy azok a lehető legkevesebb kötött időmennyiséget használják fel. Ezzel csak az a gond, hogy a korábbi cél is ez volt: a rendszerhívásokat olyanra írták, hogy lehetőleg gyorsan térjenek vissza a felhasználótérbe, s lám, mégis akadnak lappangási gondok. Vannak ugyanis olyan algoritmusok, amelyek egyszerűen nem méretezhetők jól.

Az áteresztőképesség próbája

15 „dbench 16” átlaga

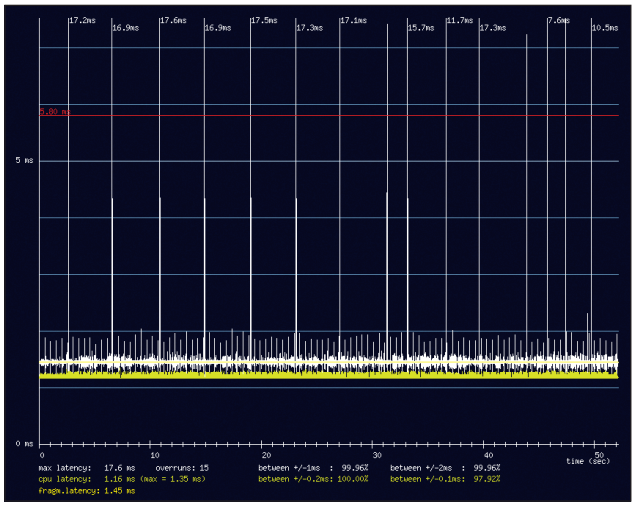
Rendszermag	Áteresztőképesség
2.5.2	24,3813 MB/mp
2.5.2-preempt	28,5920 MB/mp

A második megoldás, hogy közvetlen módon időzítőpontokat szúrunk a rendszermagba. Ezen elképzelés szerint (melynek ötlete a kis lappangási foltokból származik), meg kell keresni a rendszermag problémás pontjait, és minden ilyen helyre be kell illeszteni egy olyan kódot, amelynek „Futni szeretne valaki? Ha igen, hadd fusson!” hatása van. Ezzel a megoldással azonban az a helyzet, hogy nemigen remélhetjük, hogy az összes lehetséges problémás helyet megtaláljuk és kijavíthatjuk. Ennek ellenére – a próbák tanúsága szerint – ezek a foltok meglehetősen jó munkát végeznek. Nekünk azonban nem gyors javításra, hanem a gond hosszútávú megszüntetésére van szükségünk.

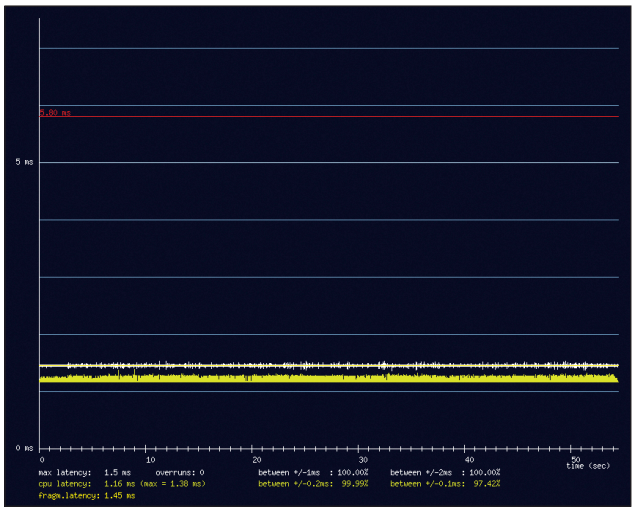
Az időosztásos rendszermag

Sokkal jobb megoldás, ha a rendszermagot időosztásosan készítjük el, hiszen ezzel a gondot teljes egészében megszüntetjük. Így ha valami fontosabbnak kell futnia, le is fog futni, függetlenül attól, hogy a jelenlegi folyamat éppen mit csinál. Az egyetlen buktató és egyben az ok, amiért Linux nem így lett már a kezdetektől megírva, az, hogy a rendszermagnak újrahívhatónak kell lennie. Szerencsére az időosztásosság gondjait a már létező SMP- (Symmetric Multiprocessing) támogatás megoldotta. Az SMP-kód előnyeit kihasználva és néhány egyszerű módosítással kiegészítve a rendszermag időosztásossá tehető.

MontaVista-i programozók mutatták be először a rendszermag időosztásos megvalósítását. Első lépésként a `spin lock` meghatározását változtatták meg úgy, hogy a „nem időosztható” tartományokat tartalmazza. `Spin lock` alatt ugyanis nincs szükségünk erre a tulajdonságra, éppen úgy, ahogy SMP alatt sem érünk el egy adott zárolt területet azonos időben több helyről. Természetesen egyprocesszoros rendszereken valójában semmi másra nem használjuk a `spin lock`-okat, csakis az időosztásosság jelzésére. Másodszor a kódot úgy módosították, hogy a kényes részeken vagy magában az ütemezőben semmiképpen ne legyen időosztásos. Végül a megszakításból való visszatérés kódfáját úgy változtatták meg, hogy a pillanatnyi folyamatot – amennyiben szükséges – ütemezze újra.



1. kép
Lappangási próba mérési eredményei hagyományos rendszermag esetén



2. kép
Lappangási próba mérési eredményei időosztásos mag esetén

UP (uniprocesszor, vagyis egyprocesszoros) módban a `spin_lock` `preempt_disable`-ként, a `spin_unlock` pedig `preempt_enable`-ként lett meghatározva. SMP módban ezen kívül a normál zárolást is elvégzik. Mit is csinálnak ezek az új eljárások?
A `preempt_disable` és `preempt_enable` egymásba ágyazható időosztásjelzők a `preempt_count`-on változtatnak, amely új `integer` a `task_struct` szerkezetekben.
A `preempt_disable` lényegében a következő:

```
+current->preempt_count;
barrier();
```

a `preempt_enable` pedig:

```
--current->preempt_count;
barrier();
if (unlikey(!current->preempt_count
&& current->need_resched))
    preempt_schedule();
```

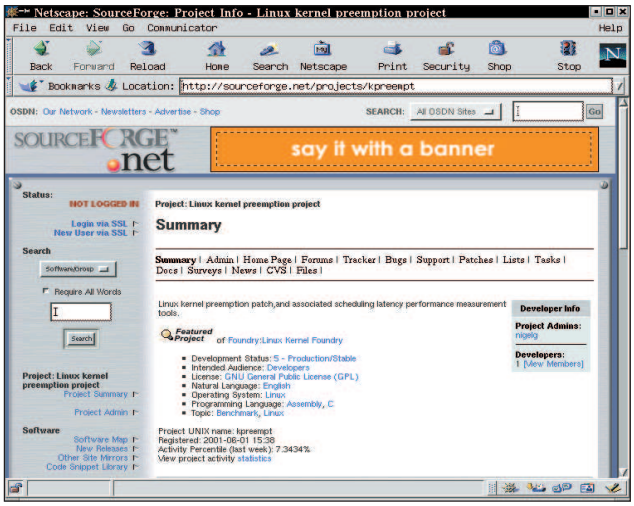
Ennek eredményképpen nem fogunk időosztást alkalmazni, amikor a számláló nullánál nagyobb, mivel a `spin lock`-ok már eleve a megfelelő helyeken vannak, hogy megóvják az SMP-gépek kritikus részeit, az időosztásos rendszermag pedig most már szintén rendelkezik ezzel a védelemmel.
A `preempt_schedule` magába az ütemezőbe való belépést valósítja meg. Beállítja a működő folyamat jelét (flag), hogy jelezze: ez a folyamat időosztásos volt, majd meghívja az ütemezőt, végül visszatéréskor leszedi a jelet:

```
asmlinkage void preempt_schedule(void)
{
    do {
        current->preempt_count +=
        ➥PREEMPT_ACTIVE;
        schedule();
        current->preempt_count -=
        ➥PREEMPT_ACTIVE;
    } while (current->need_resched);
}
```

A `preempt_schedule` másik bejárata a megszakítások visszatérő ösvényén keresztül vezet. Amikor a megszakításkezelő visszatér, ellenőrzi a `preempt_count` és a `need_resched` változókat, ugyanúgy, ahogy a `preempt_enable` is teszi (bár az ***entry.S***-ben található megszakítás visszatérő kód assembly nyelven íródott). Az lenne az eszményi, ha itt is időosztást idéznénk elő, hiszen egy megszakításról van szó, amely a `need_resched`-et általában valamilyen eszközesemény miatt állítja be. A megszakítást sajnos nem mindig tudjuk azonnal időosztásossá tenni, hiszen még zárolva lehet. Ez az oka annak, hogy az időosztás állapotát a `preempt_enable`-ben is ellenőrizzük.

Eredmények

Ezzel az időosztásos rendszermagfolttal a feladatokat nemcsak akkor ütemezhetjük újra, ha a felhasználótérben vannak, hanem azonnal, amint futtatni kell őket. Milyen hatása lesz mindennek?
A folyamatszintű válaszidő néhány esetben akár a huszadára is csökkenhet (lásd az *1. képen* bemutatott hagyományos rendszermagot szemben a *2. képen* látható időosztásos rendszermaggal). Ezek a grafikonok ***Benno Senoner*** igen hasznos lappangási próbaprogramjával készültek, amely a terhelés alatti audioátmenetitár-kezelést utánozza. Az ábra vörös vonala azt a lappangási értéket jelzi, amely felett az audiokihagyások már az emberi fül számára is észrevehetők. Látható, hogy az 1. képen több tüske is megfigyelhető szemben a 2. kép folyamatos, alacsony grafikonjával.
A lappangási próba szerint tehát a legrosszabb és az átlagos lappangás tekintetében egyaránt javulás mutatkozik. A további próbák kimutatták, hogy a rendszer átlagos lappangási ideje különféle terheléstípusok mellett most az 1–2 ms-os tartományba esik.
A leggyakoribb ellenvetés az időosztásos rendszermagokkal szemben a növekvő összetettség. Az összetettség pedig, mondják az ellenzők, csökkenti az áteresztőképességet. Szerencsére az időosztásos rendszermagfolt sok esetben még az áteresztőképességet is növeli (lásd az *1. táblázatot*). Ennek elméleti háttere az, hogy amikor az I/O-adat elérhetővé válik, az időosztásos rendszermag az I/O-hoz kapcsolt folyamatot sokkal gyorsabban életre tudja hívni. Az eredmény kellemes ráadásaként az áteresztőképesség is magasabb lesz. A főbb eredmények:



simábban futó grafikus felület, terhelés alatt kisebb audiokihagyások, jobb alkalmazás-válaszidő, és sokkal igazságosabb időosztás a magas elsőbbségű (priority) folyamatok számára.

A változások programozói szemszögből nézve

A rendszermagbetörők (hacker) valószínűleg elgondolkodnak rajta: hogyan hat mindez a kódomra? Mint korábban már említettük, az időosztásos rendszermag a már létező SMP-támogatáson alapul. Ez teszi lehetővé, hogy az időosztásos rendszermagfolt aránylag egyszerű legyen, és hogy a kódolási módszerekre gyakorolt hatása viszonylag kicsi maradjon. Egy változtatás azonban mindenképpen szükséges: jelenleg a processzoronkénti adatok (olyan adatok, amelyek minden processzor esetében egyediek) nem igényelnek zárolást. Mivel minden processzor esetében egyediek, a másik processzoron futó folyamat nem ronthatja el az első adatait. Az időosztás használatával azonban egyazon processzoron futó folyamatok is időosztássossá tehetők, s ilyenkor a második folyamat belegázolhat az első adataiba. Ez ellen normál esetben a meglévő SMP-zárak védik, de a processzoronkénti adatoknak nincs szükségük ilyen zárra. Azokat az adatokat, amelyek nem rendelkeznek zárral, mert természetüktől fogva védettek, „implicit módon zároltnak" nevezzük. Az implicit módon zárolt adatok és az időosztás nem fér meg együtt. A megoldás szerencsére egyszerű: az adat elérése körül ki kell kapcsolni az időosztást, például:

```
int catface[NR_CPUS];
preempt_disable();
catface[smp_processor_id()] = 1;   /* CPU
➥szerint indexelj k a  catface-t */
/* mßveletek a catface-en */
preempt_enable();
```

A jelenlegi Preemption-folt már tartalmazza a meglévő implicit módon zárolt adatok elérésének megfelelő módosítását. Szerencsére az ilyesmi viszonylag ritka. Azonban minden új kódnak védelemre van szüksége, amennyiben időosztásos rendszermagban akarjuk őket használni.

Továbbfejlesztési lehetőségek

Maradt még néhány feladatunk: miután a rendszermagot időosztássossá tettük, hozzáláthatunk a sokáig tartó zárolások futásidejének csökkentéséhez. Mivel a rendszermag nem lehet időosztásos, amíg a zár zárva van, a leghosszabb zárolások ideje fogja meghatározni a rendszer legrosszabb lappangási idejét. Minden munka, ami jótékony hatással van az SMP-méretezhetőségre (a finomabb felbontású zárolásra) egyúttal a lappangási időt is csökkenteni fogja. Megpróbálhatjuk újraírni az algoritmusokat és a zárolási szabályokat, csökkentve ezáltal a zárolási időt. A BKL eltüntetése szintén segíthet.
A hosszú ideig tartó zárolásokat felderíteni legalább olyan nehéz, mint újraírni őket. Létezik azonban egy `preempt-stats` folt, amely a nem időosztásos időszakokat méri és okait jelenti. Ez az eszköz igen hasznos lehet, ha adott terhelés mellett (például a Quake alatt) akarjuk meghatározni a lappangás okát. Amire szükség van, azt el kell érni. A rendszermagfejlesztőknek minden olyan zárolási időt újra meg kell vizsgálniuk, ami egy adott határértéket túllép. Egy viszonylag korszerű rendszeren ez körülbelül 5 ms-ot jelent. Ezt észben tartva rámutathatunk a magas lappangási és zárolási idejű területekre és kijavíthatjuk őket.

Összegzés

A Linux-közösség népes és igen sokrétű, a Linux felhasználási területe pedig a beágyazott rendszerektől egészen a nagykiszolgálókig terjed. Az időosztásos rendszermagmódszer előnyei a valós idejű alkalmazások területén is túlmutatnak.
Az asztali felhasználók, a játékosok és a multimédiafejlesztők egyaránt élvezhetik a csökkentett lappangási idő előnyeit.
A 2.4 és a 2.5 rendszermagfákhoz megoldás egyaránt szükséges – és elképzelhető, hogy nem a legszerencsésebb, ha mindkét változathoz azonos megoldást készítenünk. Mivel a 2.5-ös még fejlesztés alatt áll, itt az ideje, hogy beillesszenek egy azonnali előnyökkel járó képességet, amely egyúttal a további fejlesztésekhez is keretként szolgálhat. Eredményül jobb rendszermagot kaphatunk.

Linux Journal május, 97. szám

Robert Love
(rml@tech9.net) matematika és számítógépes tudományok szakos hallgató a Floridai Egyetemen. Amikor éppen nem Linuxot elemez, autóversenyzik, thai ételeket eszik vagy punk zenét hallgat.

Kapcsolódó címek

Benno Senoner lappangáspróbája
➲ http://www.gardena.net/benno/linux/audio

dbench ➲ http://ftp://samba.org/pub/tridge/dbench

Időosztásos rendszermagfolt
➲ http://www.kernel.org/pub/linux/kernel/people/rml/preempt-kernel

A Preemptive Kernel Project honlapja
➲ http://kpreempt.sourceforge.net

Alacsony lappangási folt
➲ http://www.zipworld.com.au/~akpm/linux/schedlat.html

MontaVista ➲ http://www.mvista.com

preempt-stats folt
➲ http://www.kernel.org/pub/linux/kernel/people/rml/preempt-stats

A Linux Capabilityk használata

A paranoiás rendszergazdák legnagyobb örömére ezúttal a felhasználók jogosultságainak korlátozásával, azon belül is a POSIX Capabilitykkel foglalkozunk.

Mostanában gyakori téma a biztonság, és erre minden okunk megvan. Minél jobban elterjednek a hálózatok és az internethasználat, a biztonság egyre fontosabbá válik. Mint minden jó rendszer, a Linux is folyamatosan fejlődik, hogy megfeleljen az egyre növekedő biztonsági elvárásoknak.
A biztonság egyik lépcsője a felhasználók jogainak megfelelő beállítása. A Unix-stílusú felhasználói jogosultságoknak két fajtája létezik: felhasználói és rendszergazdai, vagyis `root` jogosultság. A felhasználók alapértelmezetten semmilyen jogkörrel nem rendelkeznek: nem szólhatnak bele más felhasználók folyamatainak működésébe, és mások fájljait sem változtathatják meg. Az alkatrészek közvetlen elérése és a legtöbb hálózati lehetőség szintén korlátozott. Ezzel szemben a rendszergazda szinte bármit megtehet.
Előfordulhatnak azonban olyan esetek is, amikor köztes megoldásra lenne szükségünk. Megeshet, hogy egy folyamatnak feladata elvégzéséhez különleges jogosultságokra van szüksége, ugyanakkor a teljes körű rendszergazdai hozzáférés túlzás. A `ping` program például `setuid-root` jogosultsággal rendelkezik, így alkalmas ICMP-csomagok küldésére. A veszély abban rejlik, hogy még mielőtt a `ping` eldobná a rendszergazdai jogosultságokat, az esetlegesen benne lévő hibákat egy rossz szándékú program kihasználhatja, és ezáltal rendszergazdai jogosultságokat szerezhet.
Szerencsére most már létezik megfelelő köztes megoldás: a POSIX Capabilityk. Ezek a Capabilityk a rendszergazdai jogosultságokat logikus csoportokra osztják, melyeket tetszés szerint adhatunk ki a folyamatoknak vagy vonhatunk meg tőlük. A Capabilityk segítségével a rendszergazda pontosan meghatározhatja, hogy egy folyamat milyen műveleteket hajthat végre, és ily módon jelentős mértékben csökkenthető a rendszert fenyegető kockázat. Ha szerencséd van, foltozásra sem lesz szükséged.
Az összes Capability listája a */usr/include/linux/capability.h* fájlban található meg, ahol a felsorolás a `CAP_CHOWN`-nal kezdődik. Az egyes sorok jelentése elég nyilvánvaló, ráadásul minden eléggé jól le van írva. Az egyes Capabilityk csak egyszerűen bitek egy bittérképen. Ez a bittérkép jelenleg 32 bit hosszú, és 28 bit már használatban áll. Jelenleg éppen arról folynak viták, hogyan bővítsék e bittérképet.

A Proc felület

Jelenleg, a 2.4.17-es rendszermag idejében a */proc/sys/kernel/cap-bound* állomány egy 32 bites integert tartalmaz, ami a pillanatnyi rendszerszintű Capability-készletet jelöli. Ez a globális Capability-készlet határozza meg, hogy a rendszeren futó programok milyen képességeket birtokolhatnak. Ha egy Capabilityt lecsippentünk, a rendszeren lévő összes folyamat képtelen lesz az ehhez a Capabilityhez kapcsolódó műveletek végrehajtására, a rendszergazdai folyamatokat is beleértve! Ha például valaki betör a rendszerünkbe, sok esetben első dolga, hogy egy úgynevezett támadócsomagot (rootkit) telepít, vagyis egy olyan eszközkészletet, amellyel elrejtheti a kiszolgálón végzett tevékenységét és tovább fertőzhet, illetve hátsó kaput építhet a rendszerbe – ezen keresztül később bármikor visszatérhet. Ezt követően olyan modult tölt be a rendszermagba, amellyel saját folyamatait leplezheti. A rendszergazda, hogy ezt megakadályozza, a rendszerindítási folyamat utolsó lépéseként a `CAP_SYS_MODULE` Capabilityt eltávolíthatja a rendszerből. Ezzel a lépéssel újabb modulok betöltése vagy eltávolítása hiúsítható meg. Ha egyszer egy Capabilityt eltávolítottunk, nincs mód az újbóli hozzáadására. Amennyiben összes képességét mégis vissza szeretnénk állítani, a rendszert újra kell indítani (ezt a `CAP_SYS_BOOT` Capability eltávolításával ugyancsak meggátolhatjuk, ezt követően a rendszert már csak magán a gépen lehet kikapcsolni).
Rendben, lódítottam. Két mód is létezik arra, hogy visszaállítsunk egy Capabilityt:

1. Az `init` elvileg képes Capabilityk újbóli felvételére – legjobb tudomásom szerint azonban ennek megvalósítása még várat magára. Erre a tulajdonságra Capabilityk kezelésére felkészült rendszerek esetén lehet szükség, amennyiben meg akarjuk változtatni a futási szintet.
2. Ha egy folyamat rendelkezik a `CAP_SYS_RAWIO` Capabilityvel, akkor a */dev/mem* állományon keresztül képes módosítani a rendszermag memóriáját, s így olyan jogosultságokat biztosíthat magának, amilyeneket csak akar. Természetesen ezt a Capabilityt is eltávolíthatjuk, de vigyázzunk, mert így egyes programok (pl.: az X) nagy valószínűséggel működésképtelen lesznek.

A `cap-bound` kézi szerkesztése elég fárasztó, szerencsére azonban létezik egy lcap nevű eszköz, mely a Capabilityk beállítását hivatott megkönnyíteni. Lássunk egy példát a `CAP_SYS_CHOWN` eltávolítására:

```
lcap CAP_SYS_CHOWN
```

Ezután már x egy fájl tulajdonosát képtelenség megváltoztatni:

```
chown nobody test.txt
chown: changing ownership of  test.txt :
➥Operation not permitted
```

A három felsorolt kivételével az összes Capability ilyen módon távolítható el:

```
lcap -z CAP_SYS_BOOT CAP_SYS_KILL
➥CAP_SYS_NICE
```

Fontos, hogy a `cap-bound` módosításával csak az újonnan induló programok jogait korlátozhatjuk, pontosabban csak azokét, amelyek az `exec(2)`-t meghívják (vess egy pillantást a rendszermag forrásában található *fs/exec.c* fájlban a `compute_creeds` függvényre).
Azok a programok, amelyek már korábban is futottak, jogaikat megtartják.
A következő részben a létező folyamatok képességeinek megváltoztatásáról és a már említett csapdáról szólunk. Amennyiben az `lcap` parancsot kapcsolók nélkül futtatjuk, kiírja, hogy jelenlegi rendszerünk milyen képességekkel rendelkezik. Ha a beállításaidban a `CAP_SETPCAP` nincs bekapcsolva, akkor rendszermagodon először végre kell hajtanod

egy apró változtatást. A rendszermag forrásában az ***include/linux/capability.h*** állományban cseréld le a következő két sort (lásd a ➲ http://www.linuxvilag.hu/capability webhelyen). Ezt követően rendszermagodat fordítsd újra.

Valójában annak is megvan az oka, hogy `CAP_SETPCAP` alapértelmezetten ne legyen engedélyezve: éles rendszeren egy beállítás biztonsági kockázatot jelenthet. Bár már létezik folt ennek önműködő végrehajtására, e cikk írása idején hivatalosan még nem került be a rendszermagba. A biztonság kedvéért, miután a próbálgatást befejezted, ezt a Capabilityt távolítsd el a rendszeredből!

A cikk írása idején a folyamatok Capabilityinek megváltoztatására két utasítás áll a rendelkezésünkre: a `capset` és `capget` rendszerhívások. Ez a későbbiek folyamán még változhat. Ha szeretnénk, hogy alkalmazásaink más rendszereken is működjenek, a libcap használata ajánlott (➲ http://www.kernel.org/pub/linux/libs/security/linux-privs/kernel-2.4)

A `capset` prototípusa:

```
int capset(cap_user_header_t header, const
➥cap_user_data_t data);
```

A `header` kapcsoló nagyon ötletes módja a folyamat kijelölésének:

```
typedef struct __user_cap_header_struct {
        __u32 version;
        int pid;
} *cap_user_header_t;*
```

Ha a `pid` értéke -1, akkor az összes jelenleg is futó folyamat módosul. Ha az érték ennél kisebb, akkor az a folyamatcsoport változik, amelynek azonosítója `pid * -1`. Az alapötlet tehát ugyanaz, mint a `kill(2)` esetén.

A `data` kapcsolóval jelölhető ki, hogy melyik Capability-készleten szeretnénk módosítani. Három közül választhatunk (lásd a ➲ http://www.linuxvilag.hu/capability webhelyen).

A `permitted` azokat a Capabilityket jelöli, amelyről a programnak tudomása van.

Az `effective`-készlet azokat a Capabilityket határozza meg, amiket a folyamat a `permitted`-készletből tényleges felhasználhat. Olyan ez, mintha egész seregnyi költővel rendelkeznél (ez a `permitted-kØszlet`), de csak néhányat választhatsz ki közülük, melyekkel védheted magad (mondjuk Alan Ginsberget, ez az `effective`-készlet).

Az `inheritable`-készlet azokat a Capabilityket jelölik, amelyeket tovább örökíthetünk az `exec(2)`-kel indított folyamatokra. A `fork(2)` függvény nem változtat a Capabilityken, a létrejövő gyermekfolyamat pontosan ugyanazokkal a képességekkel rendelkezik, mint a szülője.

Csak azokat a Capabilityket vehetjük fel egy folyamat `effective`- és `inheritable`-készletébe, melyeket a `permitted`-készlet is tartalmaz. A `permitted`-készlet csak akkor módosítható, ha a folyamat rendelkezik a `CAP_SETPCAP` Capabilityvel.

Sajnálatos módon a Capabilityk egyelőre semmilyen fájlrendszert nem támogatnak, így használatuk ezen a területen nagyon korlátozott. De eljön majd az idő, amikor a rendszermagok képesek lesznek rá, hogy a program fájlleírójában (inode) tárolják az ahhoz tartozó Capabilityket, így véve elejét a `setuid` biteknek, amelyeket oly sok rendszerprogram igényel.

Ha ez működni fog, a `ping` parancsot ilyen egyszerűen lehet majd feljogosítani arra, hogy alacsony szintű foglalatokat használhasson:

```
chattr +CAP_NET_RAW /bin/ping
```

Sokkal sürgetőbb gondok miatt ennek kidolgozására sajnos még nem jutott idő.

Ha kedved van hozzá, a libcapet kedvenc szolgáltatásaid jogainak megnyirbálására is használhatod, és csak azokat a Capabilityket kapják meg, amikre tényleg szükségük van. Az `xntpd`-hez jó néhány ilyen folt létezik; néhány ekképpen módosított változathoz még külön rpm-csomag is létezik. A Google-lal rákereshetsz kedvenc programjaid Capabilitykre felkészült változataira, amelyekről úgy érzed, hogy a legnagyobb lyukat jelentik rendszered biztonságán.

A `setpcap` rendszerhívással egy már futó folyamat képességeit változtathatod meg. Ha például egy általános héj (shell) PID-je 4235, akkor ily módon ruházhatod fel azzal a képességgel, hogy minden folyamatnak képes legyen jelzést küldeni:

```
setpcaps ·cap_kill=ep· 4235
```

Tételezzük fel, hogy egy barátunk kipróbált egy CGI-programot, és az őt futtató Apache állandóan végtelen ciklusba keveredik – ekkor biztosíthatjuk számára, hogy a megbolondult Apache-okat kilője. Ezt elegendő egyszer beállítanod a bejelentkezési héjra, utána el is feledkezhetsz róla.

Most pedig azt szemléltetjük, hogy az `execcap` és `sucap` felhasználásával hogyan futtathatjuk a `ping`-et `nobody`-ként, egyedül a `CAP_NET_RAW` képességet engedélyezve számára. Pingünk célpontjául a válasszuk ➲ http://www.yahoo.com-ot:

```
execcap ·cap_net_raw=ep· /sbin/sucap nobody
➥nobody /bin/ping www.yahoo.com
```

Ez a példa nem különösebben hasznos, lévén futtatásához rendszergazdai jogosultságokkal kellene rendelkezned, de nagyszerűen megmutatja a lehetőségeidet. Ezektől a hátrányoktól eltekintve ezt az eszközt a rendszergazdák rendszerük biztonságának növelésére nagyon jól használhatják. Például a `CAP_SYS_BOOT`, `CAP_SYS_RAWIO` és a `CAP_SYS_MODULE` nélkül futó rendszer rendkívül megnehezíti a támadó dolgát. Nem nyúlhatnak bele a rendszermag memóriájába, nem indíthatnak új modulokat, de még a rendszert sem indíthatják újra, hogy egy hátsó kapukkal megtűzdelt rendszermagot betöltsenek.

Ha rendszered naplófájljai `'append-only'`-ra vannak beállítva (vagyis csak újabb sorokat lehet hozzájuk fűzni), rendszerprogramjaid pedig `'immutable'`-re (vagyis megváltoztathatatlanra), ráadásul még a `CAP_LINUX_IMMUTABLE` Capabilityt is eltávolítottad, a behatoló képtelen lesz eltüntetni a nyomait, illetve a saját módosított rendszerprogramjait sem tudja feltelepíteni. A forgalomelemző eszközök – mint amilyen a `tcpdump` is – használhatatlanná válnak, ha a `CAP_NET_RAW`-képességet eltávolítjuk. Ezenkívül a `CAP_SYS_PTRACE`-t is tüntesd el, ezzel programjaid nyomkövetését is letiltod. Az ilyen barátságtalan rendszerek a kódtörő kölykök (script kiddie) rémálmát jelentik, egyetlen lehetőségük, hogy lekapcsolódnak a rendszerről, és megvárják, amíg felfedezik őket.

Összegzés

Ezekkel a képességekkel a nagyon bonyolult szolgáltatásokat is finomhangolhatjuk, így rendszerünk biztonságát minden szempontra kiterjedően testreszabhatjuk. Ha más nem is, de a paranoiás rendszergazdák olyan eszközhöz jutnak, mely megkönnyíti a harcot az „ellenük” vívott véget nem érő küzdelemben.

Linux Journal május, 97. szám

Michael Bacarella
(mike@bacarella.com) a Netgraft cég elnöke, mely webes rendszerek fejlesztésével és biztonsági elemzéssel foglalkozik. New Yorkban él csodálatos menyasszonyával és a Kang nevű, legrémisztőbb iguánával.

Segítség a bajban: User-Mode Linux

A rendszermagtérben való programozás mindig is a guruk szakterülete volt. Kevés embernek van elegendő bátorsága, tudása és türelme a megszakítások, az eszközök és a sötét árnyként lebegő „magpánik” világában.

Amennyiben felhasználói térben írunk programot, a legrosszabb, ami történhet, hogy programunk „coredumpol”, vagyis leáll és memórialenyomata a lemezre kerül. Ilyenkor programunk valamit nagyon rosszul csinált, ezért az operációs rendszer úgy döntött, hogy a program által lefoglalt teljes memóriát és állapotadatot *core* fájl formájában visszaadja nekünk. A *core* fájlt ezután felhasználhatjuk a hibakereséshez, és kijavíthatjuk a problémát.
Amikor azonban a rendszermagot programozzuk, nincs semmilyen operációs rendszer, amely közbeléphetne és biztonságosan leállíthatná a programunkat, majd megmondaná, hogy mi volt a gond. Igaz, a Linux-rendszermag elég „rendes” saját kódjához, ugyanis néha még a pánikot is képes túlélni, amennyiben az elkövetett hiba viszonylag kicsi (ezeket a pánikokat általában „oops”-oknak nevezik). Ugyanakkor programunkat semmi sem akadályozhatja meg, hogy felülírja vagy elérje a rendszermag címterében elhelyezkedő bármely memóriaterületet. Másrészt ha a modul lefagy, a rendszermag is így tesz (gyakorlatilag a pillanatnyi rendszermagszál fagy le, azonban az eredmény általában ugyanaz).
Ezek a nehézségek esetleg nem tűnhetnek túl veszélyesnek, pedig komolyan kell vennünk őket. Ha a rendszermag pánikol, a legritkább esetben tudjuk meg, hogy tulajdonképpen mi is okozta. Gyakori megoldás, hogy `printk`-kat pakolunk mindenhová, és reménykedünk, hogy belebotlunk a hibába, mielőtt az üzenetek újraindításkor eltűnnének. Mindez természetesen feltételezi, hogy a fájlrendszert nem károsítottuk. Egyszer már elvesztettem a teljes fájlrendszeremet egy rosszkor érkező rendszermagpánik során (és főként amiatt, hogy egy rosszul meghatározott mutató az `ext2` belső adatszerkezeteinek egy részét felülírta).
Az első dolog, amit nem árt megjegyezni, ha rendszermag-programozásra adjuk fejünket: minden forrást tartsunk NFS-en. A másik gépen biztonságban maradnak a fájlok. Azonban ez sem ment meg minket attól, hogy minden egyes pánik után futtatnunk kelljen az `e2fsck`-t. Ráadásul így is elveszthetjük fájlrendszerünket, még ha a forráskód a másik gépen biztonságban is van.
Így aztán talán nem meglepő, hogy milyen kevesen vágnak bele a rendszermag programozásába. Most azonban mindez megváltozhat.

Virtuális gépek és az UML

Réges-régen, a nagygépes időkben, amikor még az időmegosztásos gépek voltak a legelterjedtebbek, megszületett a virtuális gép gondolata. A virtuális gép egy olyan beágyazott számítógép, amely teljes egészében a rendelkezésünkre áll. A virtuális gép fizikai alkatrészeket közvetlenül nem érheti el. A vassal való összes kapcsolatot a számítógép vagy egy emulátor ellenőrzi.
A VMware (➲ http://www.vmware.com) például egy meglehetősen hatékony virtuális gépet készített, amelyen minden x86-alapú operációs rendszer futtatható Windows NT, 2000, XP vagy Linux alatt. A SoftPC (egy 8086 emulátor, amely Windows- és DOS-programok futtatását teszi lehetővé) Motorola 68 k-alapú számítógépeken (vagyis Macintoshon) 1988 óta használható.
A valódi virtuális gépek néha túlságosan drágák lehetnek az érdeklődők költségvetéséhez képest (a VMware Workstation for Linux ára a honlapjukon közzétettek alapján: 299 dollár). Szerencsére immár azok számára is létezik ingyenes lehetőség, akik Linux alatt szeretnének dolgozni: a User-Mode Linux (UML).
A User-Mode Linux (➲ http://user-mode-linux.sourceforge.net) nem teljes értékű virtuális gép. Nem emulálja a különböző alkatrészeket, és nem teszi lehetővé, hogy más operációs rendszereket futtassunk. Megengedi viszont, hogy a rendszermagot a felhasználói térben futtassuk. A fejlesztés során számos előnyre tehetünk szert ezáltal: a gazdagép fájlrendszere védett marad, a virtuális fájlrendszer visszavonhatatlan (ez a károsodástól óvja meg), több gépet is futtathatunk egyetlen számítógépen (ez különösen gépközi kapcsolattartás, például hálózati üzenetek ellenőrzése során lehet hasznos, hiszen nem kell több számítógépet használnunk), illetve a rendszermagot igen könnyen futtathatjuk a hibakeresőben.

Az UML beállítása

Az UML futtatása egyszerű: valamelyik bináris csomagot (rendszermagbináris és néhány további eszköz) avagy a rendszermagfoltot is letölthetjük. Ezenkívül szükségünk lesz még a fájlrendszerre is. Azt javaslom, először játszunk el a binárisokkal, és csak azután fogjunk egy olyan saját rendszermag összeállításába, amely igényeinket jobban kielégíti. A HOWTO ezeket a témaköröket bővebben taglalja.
Az UML egyik nagy előnye a „Copy-on-Write” fájlok alkalmazásában rejlik. Ezek a fájlok lehetővé teszik, hogy a virtuális fájlrendszert az alapfájlrendszer módosítása nélkül változtassuk meg. A fájlrendszeren végrehajtott minden írási művelet vagy módosítás ezekbe az (általában *.cow* végződésű) fájlokba kerül. Így ha munka közben sikerült pánikba kergetni a fájlrendszert, mindössze annyit kell tennünk, hogy töröljük a *.cow* fájlt (ez újra létre fog jönni), és meghibásodott fájlrendszerünk máris eredeti állapotában tért vissza (léteznek olyan eszközök is, amelyekkel a *.cow* fájlban tárolt változásokat az eredeti fájlrendszerbe dolgozhatjuk be – abban az esetben, ha meg szeretnénk tartani őket).

Hibakereső modulok

Most, hogy az UML már fut, ideje alkotnunk valamit. Kipróbálás céljából írtam egy nagyon egyszerű rendszermagmodult. Négy eszközt használ: */dev/gentest[0-3]*. A modul minden eszközt egy kicsit másképpen kezel. Az első eszköz a süllyesztő (mint a */dev/null*). A második későbbi felhasználásra tárol karaktersoroza-

```
GNU gdb 5.0rh-5 Red Hat Linux 7.1
Copyright 2001 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux"...
0xa00d5631 in __kill ()
Breakpoint 1 at 0xa000e36b: file panic.c, line 52.
Breakpoint 2 at 0xa00c867e: file user_util.c, line 104.
Breakpoint 3 at 0xa00035bf: file init/main.c, line 553.

Breakpoint 3, start_kernel () at init/main.c:553
553             printk(linux_banner);
(gdb) c
Continuing.
```

1. kép Az UML futtatása GDB alatt

```
(gdb) c
Continuing.

Program received signal SIGINT, Interrupt.
0xa00e8f91 in __libc_nanosleep ()
(gdb) print *module_list
$1 = {size_of_struct = 84, next = 0xa0158720, name = 0xa3008b00 "genTest",
  size = 3664, uc = {usecount = {counter = 0}, pad = 0}, flags = 1,
  nsyms = 10, ndeps = 0, syms = 0xa3008e00, deps = 0x0, refs = 0x0,
  init = 0xa3008054, cleanup = 0xa3008078, ex_table_start = 0x0,
  ex_table_end = 0x0, persist_start = 0x0, persist_end = 0x0, can_unload = 0,
  runsize = 0, kallsyms_start = 0x0, kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "",
  kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb) printf "0x%08x\n", (int)module_list + module_list->size_of_struct
0xa3008054
(gdb) add-symbol-file ~/work/kHacking/genTest/genTest.o 0xa3008054
add symbol table from file "/home/chaos/work/kHacking/genTest/genTest.o" at
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb)
```

2. kép Modullista

```
  runsize = 0, kallsyms_start = 0x0, kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "",
  kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb) printf "0x%08x\n", (int)module_list + module_list->size_of_struct
0xa3008054
(gdb) add-symbol-file ~/work/kHacking/genTest/genTest.o 0xa3008054
add symbol table from file "/home/chaos/work/kHacking/genTest/genTest.o" at
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb) print *module_list
$2 = {size_of_struct = 84, next = 0xa0158720, name = 0xa3008b00 "genTest",
  size = 3664, uc = {usecount = {counter = 0}, pad = 0}, flags = 1,
  nsyms = 10, ndeps = 0, syms = 0xa3008e00, deps = 0x0, refs = 0x0,
  init = 0xa3008054 <init_module>, cleanup = 0xa3008078 <cleanup_module>,
  ex_table_start = 0x0, ex_table_end = 0x0, persist_start = 0x0,
  persist_end = 0x0, can_unload = 0, runsize = 0, kallsyms_start = 0x0,
  kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "",
  kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb)
```

3. kép Szimbólumfájl betöltése

tokat. A modul állapotát a harmadik eszközről olvashatjuk ki, végül a nullás eszköz a másik három eszköz bármelyike lehet aszerint, hogyan állítottuk be (a beállítást `ioctl`-hívásokkal módosíthatjuk). A rendszermagmodult megtalálhatjuk a 34. CD Magazin/UML könyvtárában, vagy a ➲ http://www.frascone.com/kHacking/gentest-0.1.tar.gz címről tölthetjük le.

Hibakeresés printkval

Nos, akkor idézzünk elő egy gonosz hibát! Tegyük fel, hogy amikor valaki megnyitja a negyedik eszközt (***cat /dev/gentest4***), a modul aljas módon végtelen ciklusba kerül: `for(;;) i++;` (lásd az 1. listát: 34. CD Magazin/UML könyvtár). A `deadlocks` leállásokat vagy fagyásokat jelent ezek gyakori hibák programírás során. Néha bizony igen nehéz felfedezni őket. Egy programozó ilyenkor egyszerűen `printk`-kat használna a hiba felderítésére: `printk("IdeØrtem!\n");`. Ez a fajta hibakeresés valóban működik, azonban a rendszer így is jó párszor kifagy, mire a hibát megtaláljuk. Folyamatos `fsck` futtatást feltételezve ez elég kellemetlen lehet. Az UML segítségével

```
This GDB was configured as "i386-redhat-linux"...
0xa00d5631 in __kill ()
Breakpoint 1 at 0xa000e36b: file panic.c, line 52.
Breakpoint 2 at 0xa00c867e: file user_util.c, line 104.
Breakpoint 3 at 0xa00035bf: file init/main.c, line 553.

Breakpoint 3, start_kernel () at init/main.c:553
553             printk(linux_banner);
(gdb) c
Continuing.

Program received signal SIGINT, Interrupt.
0xa00e8f91 in __libc_nanosleep ()
(gdb) printf "0x%08x", (int)module_list + module_list->size_of_struct
0xa3008054(gdb) add-symbol-file ~/work/kHacking/genTest/genTest.o 0xa3008054
add symbol table from file "/home/chaos/work/kHacking/genTest/genTest.o" at
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb) print *module_list
$1 = {size_of_struct = 84, next = 0xa0158720, name = 0xa3008b00 "genTest",
  size = 3664, uc = {usecount = {counter = 0}, pad = 0}, flags = 1,
  nsyms = 10, ndeps = 0, syms = 0xa3008e00, deps = 0x0, refs = 0x0,
  init = 0xa3008054 <init_module>, cleanup = 0xa3008078 <cleanup_module>,
  ex_table_start = 0x0, ex_table_end = 0x0, persist_start = 0x0,
  persist_end = 0x0, can_unload = 0, runsize = 0, kallsyms_start = 0x0,
  kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "éu\aJG\030kkkk\213Ué\212\002<;t\t\204Ât\téØþkkB\211
Ué\213ué\200>", kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb) break gOpen
Breakpoint 4 at 0xa30082ce: file gentest.c, line 254.
(gdb) c
Continuing.

Breakpoint 4, gOpen (Inode=0xa22ae640, fp=0xa2360620) at gentest.c:254
254             printk("%s: (%d:%d) open(%p, %p)\n",
(gdb)
```

4. kép Töréspontok beállítása

```
add symbol table from file "/home/chaos/work/kHacking/genTest/genTest.o" at
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb) print *module_list
$2 = {size_of_struct = 84, next = 0xa0158720, name = 0xa3008b00 "genTest",
  size = 3664, uc = {usecount = {counter = 0}, pad = 0}, flags = 1,
  nsyms = 10, ndeps = 0, syms = 0xa3008e00, deps = 0x0, refs = 0x0,
  init = 0xa3008054 <init_module>, cleanup = 0xa3008078 <cleanup_module>,
  ex_table_start = 0x0, ex_table_end = 0x0, persist_start = 0x0,
  persist_end = 0x0, can_unload = 0, runsize = 0, kallsyms_start = 0x0,
  kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "",
  kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb) c
Continuing.

Program received signal SIGINT, Interrupt.
0xa3008314 in gOpen (Inode=0xa229ca00, fp=0xa2360620) at gentest.c:269
269             for (;;) i++;
(gdb) list
264
265         // Hang when device 4  is opened
266         if (MINOR(Inode->i_rdev) == 4) {
267             int i;
268             printk("Computing pi to the last decimal position . . .\n");
269             for (;;) i++;
270         }
271
272         return 0;
273     } // gOpen
(gdb) where
#0  0xa3008314 in gOpen (Inode=0xa229ca00, fp=0xa2360620) at gentest.c:269
#1  0xa00308b6 in chrdev_open (inode=0xa229ca00, filp=0xa2360620)
    at devices.c:153
#2  0xa005f02f in devfs_open (inode=0xa229ca00, file=0xa2360620) at base.c:2739
#3  0xa002f8a7 in dentry_open (dentry=0xa22eecc0, mnt=0xa08853a0, flags=32768)
    at open.c:688
#4  0xa002f7ab in filp_open (filename=0xa2660000 "/dev/gentest4", flags=32768,
    mode=0) at open.c:646
#5  0xa002faef in sys_open (filename=0x9ffffe7d "/dev/gentest4", flags=32768,
    mode=0) at open.c:788
#6  0xa00c58b3 in execute_syscall (regs=
        {regs = {2684354173, 32768, 0, 1073819696, 1074649107, 2684353628, 42949
67258, 43, 43, 0, 0, 5, 1074419060, 35, 518, 2684353580, 43}})
    at syscall_kern.c:410
#7  0xa00c599e in syscall_handler (unused=0x0) at syscall_user.c:70
(gdb)
```

5. kép Fagyáspróba

egyszerűen csak beírjuk a `printk`-kat, és mindig új fájlrendszert indítunk a kipróbáláshoz.

Az UML a `printk`-kal segített ezt a hibát megtalálni, de ez a hiba igazából néhány újraindításnál komolyabb bosszúságot nem okozott volna. Itt az ideje, hogy elkészítsük első igazán gonosz hibánkat. Tegyük fel, hogy valaki az ötös eszközről olvas (azaz ***cat /dev/gentest5***), és a modul elkezdi a teljes memóriát felülírni: `memset(0, 0, 0xffffffff);` (lásd *a 2. listát*). A memória felülírása elég általános hiba a C programokban. Rendszermagban különösen kellemetlen, és néha azonnali újraindítást eredményezhet, így a kiadott `printk`-kból az égvilágon semmit sem látunk. Ezeket a hibákat is fel lehet `printk`-kal deríteni, csakhogy ez igen időigényes feladat.

Hibakeresés GDB-vel

Mint korábban említettem, az UML kitűnő hibakereső eszköz. Biztonságban tarthatjuk fájlrendszerünket, mialatt hibát keresünk a modulokban, de még ennél is többet tud, hiszen itt van nekünk a GDB.

A legtapasztaltabb rendszermag-programozók tudják, hogy létezik egy mód a rendszermag hibakeresésére GDB-vel, mégpedig a soros vonalon keresztül. Csakhogy – legalábbis tapasztalataim szerint – ez nem működik valami jól. A GDB csak pislákol a rendszermagban, néha ki is fagy, ráadásul mindehhez

```
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb) c
Continuing.

Breakpoint 1, panic (
    fmt=0xa0137ee0 "Kernel mode fault at addr 0x%lx, ip 0x%lx") at panic.c:52
52              bust_spinlocks(1);
(gdb) where
#0  panic (fmt=0xa0137ee0 "Kernel mode fault at addr 0x%lx, ip 0x%lx")
    at panic.c:52
#1  0xa00c6bac in segv (address=0, ip=2734719432, is_write=2, is_user=0)
    at trap_kern.c:94
#2  0xa00c7800 in segv_handler (sig=11, sc=0xa22abbc0, usermode=0)
    at trap_user.c:400
#3  0xa00c7992 in sig_handler (sig=11, sc=
      {gs = 0, __gsh = 0, fs = 0, __fsh = 0, es = 43, __esh = 0, ds = 43, __dsh
= 0, edi = 0, esi = 1024, ebp = 2720710320, esp = 2720710296, ebx = 2720623584,
edx = 0, ecx = 1073741823, eax = 0, trapno = 14, err = 6, eip = 2734719432, cs =
 35, __csh = 0, eflags = 66178, esp_at_signal = 2720710296, ss = 43, __ssh = 0,
fpstate = 0xa22abc18, oldmask = 0, cr2 = 0}) at trap_user.c:459
#4  <signal handler called>
#5  0xa30081c8 in gRead (fp=0xa2354620, userBuffer=0x804b220 "", bufSiz=1024,
    offset=0xa2354640) at /usr/src/uml/linux/include/asm/arch/string.h:488
#6  0xa0030096 in sys_read (fd=3, buf=0x804b220 "", count=1024)
    at read_write.c:162
#7  0xa00c58b3 in execute_syscall (regs=
      {regs = {3, 134525472, 1024, 1024, 134525472, 2684353564, 4294967258, 43
, 43, 0, 0, 3, 1074419252, 35, 514, 2684353516, 43}}) at syscall_kern.c:410
#8  0xa00c599e in syscall_handler (unused=0x0) at syscall_user.c:70
(gdb) frame 5
#5  0xa30081c8 in gRead (fp=0xa2354620, userBuffer=0x804b220 "", bufSiz=1024,
    offset=0xa2354640) at /usr/src/uml/linux/include/asm/arch/string.h:488
488                     default: COMMON("\n\tstosw\n\tstosb"); return s;
(gdb)
```

6. kép Memóriafelülírás

két gépre is szükségünk van. Sikeresen kerestem rendszermag-hibákat oly módon, hogy a VMWare alatt futó virtuális gép soros kapuját egy fájlba irányítottam át, azonban így is elég nehézkes volt, mivel a GDB rendszermagrésze néha kifagyott. Az UML segítségével mindez a már múlté: az UML lehetővé teszi, hogy a teljes virtuális gépet GDB-be helyezzük, és futás közben vagy akár a pánik után is csatlakozzunk. A UML GDB alatti elindításának legegyszerűbb módja, ha indításkor megadjuk a `debug` parancssori kapcsolót. Az UML ezután a GDB-t egy Xterm-ablakban elindítja, majd megállítja a rendszermagot. Ilyenkor a rendszermag indulásának folytatásához általában egyszerűen csak ***c***-t szokás ütni (lásd *1. képünkön*).

A modul hibakereséséhez először be kell tölteni a modult, aztán meg kell mondani a GDB-nek, hol találja a szimbólumfájlt, végül beállíthatjuk a szükséges töréspontokat.

Menjünk sorban: először töltsük be a modult. A forráskódhoz mellékelve találunk egy `loadModule` nevű egyszerű héjprogramot, amely betölti a modult, és amennyiben még nem léteznének, az eszközöket is elkészíti.

A modul betöltése után a GDB-ablakban nyomjuk le a CTRL-C billentyűket, ezzel a rendszermagot megállítjuk, majd nézzük meg a `module_list` mutatót. Az utoljára betöltött modulnak a lista elején kell lennie. A modul címét egy egyszerű `printf` paranccsal lekérhetjük. Erre szükségünk is lesz majd, amikor a szimbólumfájlt betöltjük (lásd *2. képünkön*).

Most töltsük be a szimbólumfájlt az `add-symbol-file MODUL_EL R SI_ T MEM RIAC˝M` paranccsal. A felhasznált fájlnév a gazdagép rendszerén értendő és nem a virtuális gépen. Miután ***y***-t válaszoltunk az „Are you sure you know what you're doing?" (Biztos, hogy tudod, mit csinálsz?) kérdésre, a szimbólumfájl betöltődik. Ha újra megnézzük a `module_list` mutatót, ellenőrizhetjük, hogy valóban helyesen töltődött-e be. Figyeljük meg, hogy az `init` és `cleanup` mutatók most már a címüknek megfelelő függvénynevekkel jelennek meg (lásd *3. képünkön*).

Attól kezdve, hogy a modult betöltöttük, tetszés szerinti töréspontot állíthatunk be. Én a megnyitáshoz állítottam be egy töréspontot, majd megpróbáltam `cat`-elni az egyik eszközt (lásd *4. képünkön*).

Futtassuk le két tesztünket, és nézzük meg, mennyire lesz nehéz meglelni a hibát, amennyiben GDB-t használunk. Az első próba során a rendszer lefagy. Csakhogy most a hibakeresőben leüthetjük a CTRL-C billentyűket, és megtekinthetjük, hol állt le. A leállaspróbában (lásd *5. képünkön*) teljesen nyilvánvaló, hogy a megállás helye a `for` cikluson belül van. Ha játszani akarunk, az `i` változó tartalmát ki is írathatjuk, hogy láthassuk, mi történik.

A memóriafelülírás egy kicsit bonyolultabb. Nem a pánik miatt, hanem mert a `memset`-et használtam. A `memset` a GNU `libc`-ben beágyazott assemblykódok beszúrásával végződik, így aztán úgy néz ki, mintha a hiba a `string.h`-ban, és nem pedig a modulunkban lenne. Ennek ellenére azért megmutatja, melyik függvényben fordult elő a hiba, így megtudhatjuk, hogy a `memset` a bűnös (lásd *6. képünkön*).

Ezenkívül a hiba felfedéséhez a pillanatnyi függvény bármelyik helyi változóját megvizsgálhatjuk (`gRead`), vagy megtekinthetjük a globális változókat.

Összefoglalás

Bár az UML egy eszközmeghajtó írásában valószínűleg nem sokat segít (mivel nem éri el a gép fizikai alkatrészeit), azért felbecsülhetetlen segítséget nyújt a rendszermagmodulok hibakeresésében. Lehetővé teszi, hogy olyan könnyedén írjunk rendszermagmodulokat, mintha bármilyen más C-programot fejlesztenénk, ráadásul pánikoktól, fagyástól vagy adatvesztéstől sem kell tartanunk. Hasznos eszköz minden rendszermag-programozó fegyvertárában.

Linux Journal május, 97. szám

David Frascone
(dave@frascone.com) jelenlegi munkahelye a SunLabs division of Sun Microsystems, Inc. Pillanatnyilag a Diameter (AAA) kivitelezése a feladata. Az IETF AAA munkacsoportjának tagja.

További érdekességek

Az O'Reilly & Associates, Inc. kiadásában 1998-ban jelent meg *Alessandro Rubini* Linux Device Drivers című könyve.

The Linux Kernel Hackers' Guide
➲ http://www.linuxdoc.org/LDP/khg/HyperNews/get

Szintén az O'Reilly & Associates, Inc., kiadásában olvasható *Daniel P. Bovet* és *Marco Cesati* Understanding the Linux Kernel 2001 című műve.

A The User-Mode Linux Kernel honlapja
➲ http://user-mode-linux.sourceforge.net

Behatolásérzékelő rendszerek

Pedro a különböző behatolásérzékelő rendszereket (IDS) veszi szemügyre, és bemutatja, hogyan is azonosítják a támadásokat.

Vajon Ön biztonságosnak tartja számítógép-hálózatát? Tudja-e, mi történik a hálózatán éppen ebben a pillanatban? Valamikor réges-régen éltek olyan hálózati rendszergazdák, akik hálózati biztonsági gondjaikra a megoldást egy egyszerű tűzfalban látták. A legutóbbi néhány évben szemtanúi lehettünk annak, hogy ez a szemlélet már a múlté. A hálózati rendszergazdáknak a furcsa jelenségekről szinte valós időben tájékoztató riasztóeszköz iránti igénye az IDS-rendszerek megjelenését eredményezte. Írásunkban az IDS-rendszerek válfajait és alapelveit mutatjuk be: a kiszolgálóalapú behatolásérzékelő rendszert (HBIDS), a hálózati behatolásérzékelő rendszert (NIDS), és egy új alapelveken nyugvó vegyes behatolásérzékelő rendszert (h-IDS). Megvizsgáljuk, hogyan értelmezhetjük, keletkezett adatokat, miként hozhatunk létre elektronikus aláírásokat a behatolásokat jelző adatmintákat, valamint a Linux-rendszeren működő behatolásérzékelő rendszereket, amilyen például a NIDS Snort-kezdeményezés.

Mit is takar a behatolásérzékelő rendszer elnevezés?

Az IDS olyan program, amely képes észlelni a szokásostól eltérő, a gépet, sőt a hálózatot is tönkretevő csomagokat és tevékenykedést. Az első IDS-rendszer még kiszolgálóalapú volt, a piacot hatalmába kerítő rendszer mégis az NIDS (hálózati behatolásérzékelő) lett. Általában mindig van egy készülék vagy program, amelyet hol érzékelőnek, hol ügynöknek neveznek. Esetenként egy vagy két hálózati kártyával felszerelt és válogatás nélküli üzemmódban dolgozik. Más szóval azt mondhatjuk, hogy ez az összes hálózati kártyára érkező csomagot befogja, és nem csak az egy bizonyos IP-címre érkezőket. Ily módon az IDS az összes hálózatba belépő csomagot ellenőrzi, megvizsgálja, hogy tartalmaznak-e gyanús karakterfüzéreket, azután dönt a megfelelő válaszlépésről: a tűzfallal folytatott párbeszéd közben új szabályokat hozhat létre az adott IP-cím kizárására, lapozási kérést vagy elektronikus üzenetet küldhet a biztonságért felelős rendszergazdának stb. Az NDIS-sel kapcsolatban fontos kérdés, hogy hová kell az érzékelőt telepítenünk: a tűzfalon belülre vagy kívülre? De hol is van az a legalkalmasabb hely? Hosszan lehetne erről vitatkozni, hiszen mindkét nézőpontnak akadnak támogatói, de kétségkívül a tűzfalon belülre és kívülre is telepített rendszer a legjobb választás.

Az IDS-ek válfajai és alapelvei

Az IDS alaposabb megértéséhez mindenekelőtt tudnunk kell, hogyan is működik. Két alapvető IDS-fajta létezik: az elektronikus aláírás, és az eltérésalapú modell. Az elektronikus aláírás avagy használatbeli visszaélést leíró modell a legáltalánosabban alkalmazott IDS-modell. Az ujjlenyomatok, illetve aláírások olyan minták, amelyek azonosítják a támadásokat a csomagban levő különféle lehetőségek alapján, úgymint a feladó címe, a célgép címe, a küldő és fogadó gép kapui, a zászlók és a hasznos teher és egyéb jellemzők alapján. Az ujjlenyomatok ezen gyűjteménye együttesen tudásadatbázist alkot, amelyet az IDS az összes csomaglehetőség összehasonlítására felhasznál, és ellenőrzi, vajon megegyezik-e valamelyik tárolt mintával. Az alábbiakban majd kitérünk a Nimda-féreg lenyomatára a Snort IDS-ben.

Az eltérésalapú modell a hálózaton belül a szokványostól eltérő viselkedést próbálja felismerni. Ennek megfelelően elsőként ki kell tapasztalnia, hogyan is működik a hálózat, azután a különböző mintákat fel kell ismernie, végül valamiféle üzenetet kell küldenie a konzolnak vagy az érzékelőnek. Az ilyenfajta IDS legnagyobb hátránya, hogy nem tudni, a hálózat a tanulási szakaszban vajon felmutatta-e az összes lehetséges viselkedésformát vagy sem. Ez a módszer ezért kezdetben nagyszámú téves riasztást eredményezhet.

A téves riasztások az IDS-től érkező jelzések, amelyek támadóról tájékoztatnak, ám valójában csak egy be nem állított változóról van szó, vagy például egy alkalmazás a szokásostól eltérő kapura küldött csomagot, és még mindig nincs szó valamiféle hátsó ajtóról. A nehézség elhárítása: a rendszergazdának időről időre figyelemmel kell kísérnie az IDS által küldött figyelmeztetéseket, majd el kell végeznie a rendszer finomhangolását.

A kiszolgálóalapú behatolásérzékelő rendszerek általában kiszolgálógépeken futnak, és csak olyan eseményeket észlelnek, amelyek kizárólag a programnak otthont adó gépre vonatkoznak. A HBIDS-rendszer fő célja a gép megóvása a külső változtatásoktól és a rossz szándékú lekérdezésektől. Az efféle IDS-ek fontosságát igazolják azoknak a rosszindulatú támadásoknak az elszaporodása, amelyek célja a weboldalak megrongálása, illetve támadócsomagok (rootkit) telepítése a számítógépekre, hogy onnan újabb gépeket fertőzzenek meg. A támadócsomagok a kalóz által a megrongálandó gépre telepített csomagok, amelyek hátsó ajtó nyitására alkalmas programcsomagokat tartalmaznak, naplóállományokat törölnek, hogy leplezzék magukat, vagy a `ps` és `nestat`-hoz hasonló parancsokat cserélik le, elrejtik a démonokat, vagy éppen kaput nyitnak meg.

Emellett a HBIDS szolgáltatásai között szerepel a támadások még megtörténtük előtti érzékelése is.

Tripwire

A Tripwire a Linux számára fejlesztett HBIDS-rendszerek egyik remek példája (lásd Linuxvilág 2001. szeptember, 34. oldal). A Tripwire HBIDS-rendszerként azonosítható, mivel állományépség-vizsgáló iránti igényünket elégíti ki. A Tripwire programmal a felhasználó meghatározhatja, a beállítóállományban pedig ki is jelölheti, hogy mely állománycsoportokat szeretné megvédeni a módosulástól, majd a Tripwire ezen állományok és állományjellemzők ellenőrzőösszegét fogja használni. Bármilyen változás esetén figyelmeztetést küld a rendszergazdának. A programmal „gyárilag” szállított beállítóállomány jó kiindulási alap, de a vakriasztások számának mérséklése érdekében a felhasználó nem mulaszthatja el a program beállítását. Fordítsunk különös figyel-

met a naplóállományokra. Nincs értelme felvenni őket a védendő állományok közé, hiszen tudjuk, hogy bármilyen esemény – mint amilyen a bejelentkezés is (login) – hatására növekednek. A Tripwire a `cron` ütemeződémonnal együtt használható különösen jól. A felhasználók ilyen módon önműködővé tehetik az ellenőrzési folyamatot, és előírhatják, hogy hol és mikor fusson.

PortSentry

A PortSentry – lásd ***Anthony Cinelli***-nek a PortSentry-t bemutató webhelyét a Linux Journal oldalán (➲ http://www.linuxjournal.com/article?sid=4751) találjuk – a Psionic Software cég Abacus-kezdeményezésének része, amely „az internetközösség számára ingyenes kiszolgálóalapú biztonsági és betörésvédelmi eszközkészlet létrehozását tűzte ki célul". A HBIDS-rendszerek egyik fontos fajtája ez, mivel képes a kiszolgálóknak címzett csomagok érzékelésére, és TCP-burkolókkal és az IP Tablesszel együtt használható. Ez az érzékelési mód hasznos, mivel a kapupásztázás gyakran a támadások előhírnöke. A PortSentry képes TCP- és UDP-pásztázások érzékelésére, kimutatva azokat a kiszolgálógépeket, ahol a jelzett szolgáltatás éppen a lekérdezett kapuban fut. A következő lépés a foltok és frissítések ellenőrzése, sőt, ha szükséges, még elérésvezérlő listákat (ACL) is létre kell hozni a TCP-burkolókkal, hogy a pásztázó kiszolgálóval mindenféle jövőbeni kapcsolatot megszakítsunk. Ezenkívül a tűzfalon is létrehozhat szabályokat, vagyis például egy olyat, hogy az IP Tables minden, a pásztázó géptől származó csomagot ejtsen el. Az alábbiakban egy PortSentry-bejegyzést mutatunk be a `syslog` rendszernaplóból:

```
Dec 9 03:03:17 mobile portsentry: [701]:
➥attackalert:
   TCP  SYN / Normal scan from host:
   200.185.61.132 / 200.185.61.132 to TCP
➥port: 111
Dec 9 03:03:17 mobile portsentry: [701]:
➥attackalert:
   Host 200.185.61.132 has been blocked via
➥wrappers
   With string "ALL: 200.185.61.132"
Dec 9 03:03:17 mobile portsentry: [701]:
➥attackalert:
   Host 200.185.61.132 has been blocked via
➥dropped
   Route using command "/sbin/iptables -I
   INPUT -s 200.185.61.132 -j DROP"
```

Swatch

A Swatch olyan rendszernapló-figyelő program, amely értesíti a rendszergazdát, amint a rendszernapló-állományban – például a */var/log/messages*-ben – felbukkan valamilyen előre megadott karakterlánc. A következő példában létrehoztam egy a Sort beállítására szolgáló egyszerű állományt, és a `snort`, valamint a `portsentry` elnevezések figyelését írtam elő. Azt is beállítottam, hogy találat esetén az eredményeket különböző színnel jelenítse meg, és a gép ekkor hangjelzést is adjon:

```
watchfor /snort/
echo red
bell
watchfor /portsentry/
echo blue
bell
```

Találat esetére a Swatchnak előírhatom, hogy hová küldjön üzenetet vagy milyen parancsot hajtson végre. Az előbbi Swatch-beállítóállomány az alábbi üzenetek megjelenését eredményezte:

```
Dec 9 03:22:53 flamengo snort[3268]:
➥[1:1256:2]
    WEB - IIS CodeRed v2 root.exe access
➥[Classification:
    Web Application Attack]: [Priority: 1]:
    {TCP} 200.31.36.11:253 ->
200.204.68.154:80
Dec 9 03:22:53 mobile portsentry [701]:
attackalert:
     TCP SYN / Normal scan from host:
     200.185.61.132 / 200.185.61.132 to TCP
port: 111
```

LIDS

A LIDS rövidítés a Linux behatolásérzékelő rendszert jelöli, amely a Linuxot – a telepített rendszerfoltok révén – kivételes szolgáltatásokkal ruházza fel.

A szolgáltatások között megtalálható az állománysértetlenség- és a folyamatvédelem, valamint a kapupásztázás-érzékelés. Az első kettő részletesebb magyarázatot igényel. Az állomány- és a folyamatvédelem még a rendszergazdával szemben is működik. Ez olyankor bizonyul rendkívül hasznosnak, amikor a kalóz egy rendszerbiztonsági rést – amilyen mondjuk egy átmeneti tár túlcsordulása – kihasználva megkaparinthatná a rendszergazdai jogosultságokat és bármit szabadon megtehetne, tehát akár támadócsomagokat is telepíthetne, naplóállományokat módosíthatna, vagy HTML-oldalakat törölhetne stb. Ezekkel a szolgáltatásokkal ACL-eket határozhatunk meg, amelyek szabályozhatják az állományokhoz való hozzáférést, jelszavakat tartalmazhatnak ezek olvasásához, illetve módosításához, a nem kívánt változások pedig elkerülhetők, származzanak azok meg nem hatalmazott felhasználótól vagy magától a rendszergazdától. Ugyanez igaz a folyamatokra is, mert ez a rendszer megakadályozza a bináris állományok, illetve a démonok megváltoztatását is. E megoldás további kellemes szolgáltatása, hogy a rendszermag területén kapupásztázást képes végezni.

NIDS

A hálózati behatolásérzékelő rendszerek az IDS-ek azon fajtái, amelyek képesek a hálózatot érintő támadásokat érzékelni. Fontos vitatéma, hogy hová is kellene telepíteni őket. Lehet olyan hálózati elrendezést találni, amelyben a tűzfal előtt találjuk őket, és elképzelhető olyan is, amikor mögötte. Mint már említettem, mindkettő mellett jó érveket lehet felsorakoztatni: csak a helyi igények dönthetik el, mikor melyik szükséges. Ezekben a példákban a nyílt forrású Snortot fogom használni.

Snort

Marty Roesch fejlesztette a Snortot, és jelenleg már ezernél is több szabály alapján érzékeli a támadásokat az egyszerű pásztázásoktól kezdve egészen a legújabb fajta `ssh CRC32` biztonsági résig – részletesebben lásd ***Nalneesh Guar*** Snort: IDS tervezése az Ön vállalkozása számára című írását a Linux Journal honlapján (➲ http://www.linuxjournal.com/article.php?sid=4668). A Snort óriási előnye, hogy az igényeknek megfelelően képes új szabályt alkotni. Ezzel szemben más IDS-szállítóknál általában meg kell várni a legfrissebb csomagok megjelenését, a támadások nyomán a rendszer testreszabható és új lenyomatok

készíthetők. A fenti esetre kiváló példa volt a `wu-ftpd` esete 2001 decemberében. Mindössze néhány órával a biztonsági rés nyilvánosságra hozása után a Snort-szűrő már hozzáférhető volt a biztonsági levelezőlistákon. A Snort képes együttműködni a tűzfalakkal is, vagyis például a Check Point FireWall-1-gyel, az OPSEC lehetőség felhasználásával, vagy a bedolgozó programok segítségével, hogy együttműködjön a Linux IP Tables programjával.
Amellett, hogy a Snort hatalmas ujjlenyomat-adatbázissal rendelkezik, és főként a használatbeli visszaélésleíró modellen alapszik, bétaszolgáltatásként az eltérésalapú modellt is felvonultatja. E szolgáltatás, amelynek a SPADE keresztnevet adták, az összegyűjtött adatok statisztikai elemzését adja, és megkísérli meghatározni a normális viselkedés jellegét. Amint az a nyílt forrású alkalmazásoknál lenni szokott, számos kapcsolódó alkalmazással használhatjuk.
Kellemes alkalmazás a Silicon Defense cégtől a SnortSnarf, amely a Snort által összegyűjtött adatokból HTML formátumú jelentéseket készít. A Snort egyetlen hálózati kártyával is megelégszik. Más NIDS-rendszerektől eltérően, amelyek két hálózati kártyát igényelnek – egyet az adatgyűjtéshez, egy másikat pedig a rendszergazdai felület számára – a Snort mindössze egy hálózati kártyával is dolgozni tud csomagválogatás nélküli üzemmódban, és a felügyeleti feladatot is képes ellátni, felvéve és frissítve az új szabályokat.

Vegyes IDS

Újabban egyre népszerűbbé válik egy másfajta elgondolás: a vegyes behatolásérzékelő rendszer. ***Marcus Ranum***, a Network Flight Recorder (NFR) alapítója úgy véli, hogy „a vegyes IDS-rendszerek a HBIDS- és NIDS-rendszerek erejének és gyengéinek érdekes vonalát képviselik". Ez azt jelenti, hogy ebben a rendszerben a NIDS-rendszer számos jellemvonása felbukkan, ám ezúttal gépenkénti alapon. Ebből több előny is származik, minthogy a kiszolgálóra irányuló behatolási kísérleteket egyediként próbálja azonosítani; az általa elemzett csomagokban csak a célgép IP-címét tartalmazó csomagok fordulnak elő. Az ilyen fajta IDS-nek nagy hátránya, hogy minden egyes kiszolgálógépre telepíteni kell.

Prelude IDS

A Prelude is a vegyes rendszerek csoportjába tartozik. Két részre osztható: a Prelude NID elnevezésű érzékelőre, amely a csomagok befogásáért és elemzéséért felelős, és a jelentéskészítőkiszolgálóra, amelyet az érzékelő a behatolási kísérlet jelzésére használ. A Prelude egy érdekes jellemvonása külön említést érdemel: képes szabályokat kiolvasni a Snort IDS-ből, más szóval kész futtatható szabálygyűjteménnyel rendelkezik. Webhelyéről bármelyik Prelude IDS-ről képes a szabályokat leolvasni. A Prelude-öt a fürtözési elvet figyelembe véve építették, ez a magyarázata annak, hogy a jelentéskészítő-kiszolgálót egy másik gépre telepítették, amely képes a begyűjtött adatokat felhasználóbarát formába önteni, például HTML-állománnyá alakítani.

A lenyomatok értelmezése és létrehozása

Mint már a cikk fentebbi részéből kiderült, a lenyomatok tulajdonképpen támadási minták. Fontos, hogy megértsük, miképpen működnek. Szükség esetén, vagy egy új biztonsági rés felfedezésekor magunk is létrehozhatunk ilyeneket. Példáink megvilágítják, hogyan kezeli a Snort az elektronikus ujjlenyomatokat. 2001 második felében új és nagyon hatékony férgek jelentek meg a Világhálón: Code Red, Code Red II és a Nimda. A támadások idején a Linux-felhasználók – jómagam is ezek egyike voltam – roppant szerencsésnek érezték magukat, mivel a férgek főleg a Microsoft Internet Information Serverehez kapcsolódtak. Ezeknek a férgeknek eltérő volt a mintája, például a Microsoft IIS-en keresztül megpróbálta elérni a ***cmd.exe*** állományt. Ennek tudatában könnyű volt megalkotni a Nimda Snort-szabályt, amint azt „Az IDS-ek válfajai és alapelvei" című részben már olvashattuk:

```
alert tcp: $EXTERNAL_NET any ->
➥$HTTP_SERVERS:80
msg: "WEB-IIS cmd.exe access"
content: "cmd.exe"; nocase
classtype:web-application-attack; sid:1002;
➥rev:2;)
```

Rendben, és mit jelent mindez? A Snort-értékek valójában két csoportba sorolt értékek sorozatát alkotják, amellyel a Snort figyelmét ráirányíthatjuk bizonyos dolgokra. Az első rész a szabályfejléc, és az első zárójelig mindenféle megtalálható benne. Az első érték azt mondja meg, mi kell tenni, ha a csomagegyezést talál. Az `alarm` üzenetriasztást fog végezni, és naplózza a csomagot. A második érték jelzi a Snortnak, hogy milyen protokollt szeretnénk használni, jelen esetben csak TCP-t. A következő öt érték mutatja a feladó IP-címét, a csomag irányát, a címzett IP-címét és adatkapuját. Ilyen módon biztosak lehetünk afelől, hogy a hálózatunkon kívül eső címről, a 80-as kapun – a webkiszolgálók rendszerint éppen ezen a kapun figyelnek – érkező csomagokat a belső szabálylehetőségek alapján fogják ellenőrizni. A szabálylehetőségek szakaszriasztási üzeneteket és a csomagok ellenőrizendő részéről adatokat tartalmaz. E vizsgálat eredményének ismeretében a megfelelő tevékenységet hajtja végre.
A példánkban szereplő szabálylehetőségek:

- `msg: WEB-IIS cmd.exe access` – üzenet, a riasztás leírása.
- `flag: A+"` (zászló) – logikai műveletjel, a csomagban levő összes zászló ellenőrzése.
- `content: cmd.exe` (tartalom) – beállítja a hasznos terhet.
- `nocase:` – a lehetőség megengedi a kis- és nagybetűk megkülönböztetésének figyelmen kívül hagyását.
- `classtype: web-application-attck` (osztálytípus: webalkalmazást érő támadás).
- `sid:1002` – `sid`, azaz Snort-azonosító.
- `rev: 2` – a szabály változatszáma.

További érdekességek

Marcus Ranum: Jelentés a behatolásérzékelő rendszerekről (IDS) ➲ http://www.intrusiondefense.net
A hálózati behatolásérzékeléssel kapcsolatos témában Stephen Northcutt, Judy Novak és Donald McLachlan: An Analyst's Handbook (A rendszerelemző kézikönyve, 2. kiadás), New Riders, 2000.
A SANS Intézet honlapja
➲ http://www.sans.org/infosecFAQ
A Silicon Defense cég webhelye
➲ http://www.silicondefense.com
A Snort webhelye ➲ http://www.snort.org

A Snort felhasználói kézikönyvben harmincnál is több lehetőség található a felhasználói igények kielégítésére. Úgy véli, túlságosan bonyolult? Á, dehogy! Tegyünk egy próbát, és jelezzük szabállyal a hálózatról pornográf oldalakra történő elérési kísérletet:

```
alert tcp: $EXTERNAL_NET any ->
➥$HTTP_SERVERS:80
msg: "Web porn access attempt"
content: "Free porn"; nocase;flags:A+);
```

A létrejött adatok elemzése

Egy szolgáltatáshoz tartozó kapupásztázás olyan, akár a `portmap` (111-es kapu), amelyről közismert, hogy számos biztonsági rést rejt magában, amit a PortSentry biztosan észrevenne.

```
Dec 9 03:03:17 flamengo portsentry [701]:
➥attackalert:
    TCP SYN / Normal scan from host:
     200.185.61.132 / 200.185.61.132 to TCP
     ➥port:111
```

A naplóállományok értelmezési képessége kulcsfontosságú, hogy a behatolást vizsgáló vagy a biztonsági szakember pontosan tudja, mi a teendője egy adott helyzetben. A PortSentry-ből származó fenti riasztás a `syslog` rendszernapló állományból származik. Ez a riasztás azt állítja, hogy december 9-én 3:03-kor a ***flamengo*** nevű kiszolgálógép, amelyre a PortSentry telepítve van, egy SYN-zászlós normális kapupásztázást talált a 111-es TCP-kapun, amely rendszerint a `portmap` szolgáltatást futtatja, ezúttal a `200.185.61.132`-es címről.

Összegzés

A tűzfal elsődleges biztonsági elem a hálózatban, azonban képtelen már megnyitott szolgáltatásokra irányuló támadást érzékelni, mint például egy DNS- vagy egy webkiszolgálóra irányuló támadást.
Az IDS egyedüli biztonsági elemként nem fogja megoldani gondjainkat: amennyiben elvégzi rendszerünk testreszabását, segíteni fog a figyelmeztetések helyes felismerésében, ha a hálózatban szokatlan tevékenység zajlik, vagy ha a kiszolgálóhoz vagy a hálózathoz illetéktelen hozzáférési kísérlet történne. Ezekkel az adatokkal – a behatolási IP-címmel együtt – már fel lehet keresni a rendszergazdát, és tájékoztatni lehet (a többi felhasználót is), hogy mi zajlik a hálózaton.

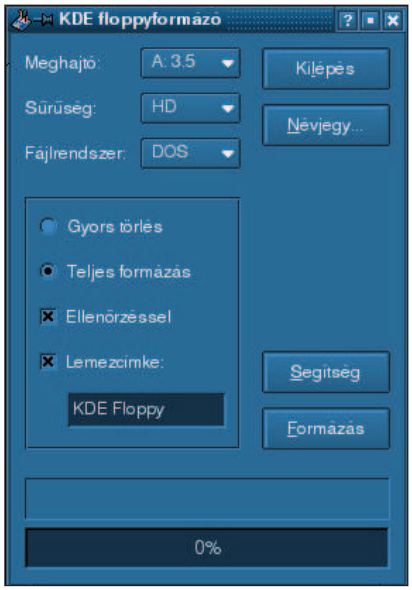
Linux Journal május, 97. szám

Pedro Bueno
(bueno@ieee.org) korábban a Lucent Technologies adattechnikai tervezőmérnöke, jelenleg az Open Communications Security biztonsági kérdésekkel foglalkozó mérnöke. Önkéntesként résztvesz a Best Linux-változat fejlesztésében. Kedvenc időtöltése a foci mellett a Snort által létrehozott riasztások megfejtése.

Fűzzünk be!

Sorozatunk e részében a Unixok egyik érdekes sajátosságával, a lemezegységein lévő fájlrendszerek be-, illetve kifűzésével foglalkozunk részletesebben.

A szóban forgó műveletet `mount`-nak hívják. Akik ez idáig kizárólag DOS-, illetve Windows-rendszereket használtak, megszokhatták, hogy az összes különálló lemezegységen helyet foglaló fájlrendszer egymástól teljesen el van választva. Ez a gyakorlatban úgy néz ki, hogy a rendszer azonosítóként minden meghajtóhoz egy betűt rendel – a későbbiekben ennek segítségével hivatkozhatunk rá. Nem nehéz belátnunk, hogy rendkívül merev rendszerről van szó, amelyben elég nehéz jól szervezett könyvtárszerkezetet létrehozni. Mivel a Unix tervezésénél kiemelt figyelmet fordítottak a rugalmasságra, a háttértárolókon található fájlok ábrázolásánál nem ezt a sémát alkalmazták. Az ötlet a következő: legyen egy úgynevezett fő (root) fájlrendszer (/), amelynek könyvtárai alá további fájlrendszereket be lehet fűzni. Így tulajdonképpen egy fájlrendszerben látjuk az összes lemezegységünkön található összes állományt és könyvtárat.



Ezzel a megoldással azonban némi gondunk akad – meg kell mondanunk a rendszernek, hogy a gépünkben található sok lemezterületből melyik látja el a fő fájlrendszer szerepét, illetve hogy mely könyvtárak alá fűzze be a többi lemezterületen lévő fájlrendszert. A Linuxszal most ismerkedő olvasóknak szánt sorozatunk mostani részében ennek módját próbáljuk bemutatni.

Rendszerünk futása során fájlrendszerünkhöz bármikor új lemezegységeket fűzhetünk, illetve a már feleslegessé váltakat kifűzhetjük. A Unix világában azt a műveletet, amelynek során új lemezegységet fűzünk be, „mountolásnak” nevezzük.

A rendszer indításakor a fő lemezterület befűzéséről maga a rendszermag gondoskodik. Azt, hogy melyik lemezterületünk legyen a fő lemezterület, az általunk használt Linux indításkezelőjében (például LILO) kell beállítanunk. Természetesen terjesztésünk telepítőprogramja erről már gondoskodott, nekünk csak akkor szükséges megbolygatnunk, ha új fő lemezterületet szeretnénk létrehozni (a Linux indításkezelőivel sorozatunk egy későbbi részében foglalkozunk).

Előfordulhat, hogy nemcsak a fő lemezterületet szeretnénk befűzni, hanem például a Linux betöltésekor a DOS/Windows lemezterületeket is önműködően be akarjuk fűzni, vagy rendszerünket több lemezterületre telepítettük, ekkor használhatjuk a */etc/fstab* fájlt. Az összes terjesztés lehetővé teszi, hogy pingvinünk ne egyetlen lemezterületen terpeszkedjen, hanem bizonyos részeit, mint például az alkalmazások binárisait tartalmazó */usr*, vagy a felhasználói könyvtárakat tartalmazó */home* könyvtárakat külön-külön lemezterületen helyezhessük el. Ez első látásra értelmetlennek tűnhet, valójában azonban nagyon sok gyakorlati haszna van, mint azt a későbbiekben látni is fogjuk.

Ahhoz tehát, hogy a rendszer indulásakor további lemezegységek is önműködően befűzésre kerüljenek, értékeiket először a */etc/fstab* fájlban meg kell adnunk. Nyissuk meg ezt az állományt! Figyeljünk arra, hogy ezt is, mint a rendszer összes beállítóállományát, csak a rendszergazda szerkesztheti. Az *fstab* felépítését vizsgálva láthatjuk, hogy minden bejegyzés külön sorban foglal helyet. Az összes terjesztés telepítője a legfontosabb dolgokat beállítja helyettünk, az intelligensebbek még a cserélhető lemezegységeket (például hajlékonylemez, CD-ROM), illetve az idegen (nem linuxos) lemezterületeket is felveszik. Ha mégsem történt volna meg, ez a feladat ránk vár. Ám előbb ismerkedjünk meg az *fstab* felépítésével!

Minden bejegyzés hat oszlopból áll. Az első oszlopba az adott lemezegység nevét kell írnunk. Sorozatunk korábbi részében már említettük, hogy a Unixban minden fizikai eszközre az úgynevezett eszközfájlok segítségével hivatkozhatunk. Ezek olyan különleges állományok, amelyek a */dev* könyvtár alatt találhatók. Minden eszközhöz egyedi eszközfájl tartozik. Minket jelenleg csak a lemezegységeket azonosító eszközállományok érdekelnek, úgyhogy most csak azokat vesszük sorra. Az első IDE vezérlőnk master csatlakozójára rákötött eszköz neve `hda`, a slavere csatlakozó pedig `hdb`. A második IDE-meghajtó esetében `hdc`, illetve `hdd`. Merevlemez esetében a rajta található lemezterületeket is azonosítanunk kell, ezt az eszköz neve után írt számmal tehetjük meg. A szabály ez esetben egyáltalán nem bonyolult: 1–4-ig az elsődleges lemezterületeket azonosítjuk (egy merevlemez legfeljebb négy darab elsődleges lemezterületet tartalmazhat), 5-től felfelé pedig a logikai tartományokat vagy lemezterületeket (partition).

Az SCSI merevlemezeknél `hd` helyett `sd`-t kell írnunk. Az első hajlékonylemezes meghajtó az `fd0`, a második az `fd1`.

A második oszlopban az úgynevezett `mount point`-ot kell megadnunk. Ez nem más, mint az a könyvtár, amely alá az adott lemezegység fájlrendszerét be szeretnénk fűzni. Figyelem! `Mount point`-nak csak létező könyvtárat adhatunk meg, tehát a befűzés előtt létre kell hoznunk egyet (például a `mkdir` paranccsal).

A harmadik oszlop a fájlrendszer típusa. Linux-lemezterület esetében ext2, illetve az újabb rendszerek esetében már ext3 vagy más típusú lemezterületet is használhatunk. Az ext3 viszonylag új (ám már a 90-es évek elejétől létező módszerre épül), a „hagyományos” fájlrendszerek működésétől gyökeresen eltérő, úgynevezett naplózó fájlrendszer. Előnye az ext2-vel szemben, hogy több kisebb blokk írása esetében gyorsabb, és kevésbé sérülékeny az áramszünet okozta leállásokra. Windowsos lemezterület esetében a `vfat`-ot használjuk (DOS esetében az

Példa a /etc/fstab állományra

```
/dev/hda8      /              ext3      defaults          1    1
none           /proc          proc      defaults          0    0
/dev/hda9      swap           swap      d efaults         0    0
/dev/cdrom     /mnt/cdrom     iso9660   noauto,ro,user    0    0
/dev/fd0       /mnt/floppy    auto      noauto,rw,user    0    0
/dev/hda1      /mnt/c         vfat      defaults          0    0
/dev/hda5      /mnt/d         vfat      defaults          0    0
/dev/hda6      /mnt/e         vfat      defaults          0    0
/dev/hda7      /mnt/f         vfat      defaults          0    0
```

msdos-t), CD-ROM-nál pedig az `iso9660` kulcsszót.
A következő oszlopban további szolgáltatásokat határozhatunk meg az egység csatolására vonatkozóan. Nézzük meg a legfontosabbakat:

- `ro`: az egységet kizárólag olvasásra fűzzük be. Az ily módon befűzött fájlrendszerbe még a rendszergazda sem írhat! Az esetleges vírustámadások megelőzéseként nem rossz ötlet, ha ilyen módon fűzzük be a binárisokat tartalmazó */usr* könyvtárat, ám új alkalmazás telepítése előtt írásra is be kell fűzni.
- `rw`: az egység írásra és olvasásra is be lesz fűzve.
  user/nouser: a rendszergazdán kívül más felhasználónak is megengedjük/megtiltjuk az egység ki- és befűzését.
- `exec/noexec`: az adott lemezegységről bináris futtatásának engedélyezése/tiltása. Jó szolgálatokat tehet a rendszergazdának, ha meg szeretné akadályozni, hogy a felhasználók saját – a /home-ban található – könyvtárukból maguk hozta programokat futtassanak.
- `suid/nosuid`: a sorozatunk előző részében bemutatott Set UID-es programok futtatásának engedélyezése/tiltása az adott egységről. Leginkább biztonsági célokat szolgál.
- `auto/noauto`: az adott egység a rendszer indulásánál önműködően be legyen-e fűzve vagy sem. Cserélhető lemezegységeknél a `noauto` kapcsolót használjuk!
- `defaults`: a következő kapcsolókat foglalja magában: `rw`, `auto`, `nouser`, `exec` és `suid`.

A maradék oszlopok értéke csak egész szám lehet. Az utolsó előttit a `dump` parancs használja annak meghatározására, hogy szükséges-e a fájlrendszerről biztonsági másolatot készíteni. Véleményünk szerint ide `0`-t érdemes írnunk.
A hatodik és egyben utolsó oszlopban adhatjuk meg, hogy a fájlrendszer-ellenőrző `fsck` program milyen sorrendben ellenőrizze a lemezterületeket. Az `1`-es az indító (boot) lemezterület. Amennyiben egy lemezterülethez `0`-t írunk, nem fog ellenőrzésre kerülni.
Listánkon szerzőnk */etc/fstab* állományát láthatjuk. E sorozat írója ugyan tisztában van a Linux-rendszer minden előnyével, ám mégsem volt szíve megválni windowsos rendszerétől, sőt, mindennek tetejébe még merevlemezének első (primary) lemezterületére telepítette (*/dev/hda1*). Teljes Linux-rendszerének csupán egyetlen logikai tartományt szentelt (*/dev/hda8*). Láthatjuk, hogy az *fstab* állomány a fő lemezterület feltüntetésével kezdődik, a második sorban pedig a csereterület (swap) jellemzőinek megadása következik (*/dev/hda9*). A csereterületet is a telepítés során kell létrehoznunk, amelyet a rendszer virtuális memóriaként kezel. Szerzőnk gépében még egy hajlékonylemez-meghajtó (*/dev/fd0*), illetve egy, a második IDE-vezérlő master csatlakozójára kötött CD-ROM (*/dev/hdc*) is található. Mivel néhány dolgot a windowsos lemezterületeken tárol, időnként szükséges, hogy ezekhez Linux alól is hozzáférjen, ám ezeket a lemezterületeket felesleges minden alkalommal önműködően befűzni. Szükség esetén a befűzést kézzel végezzük (lásd a későbbiek folyamán).
Szemfülesebb olvasóink biztos felfigyeltek már a zavarbaejtő második sorra, ahol `mountpoint`-ként a */proc* van megadva. Ez nem valódi, hanem virtuális fájlrendszer, amelyet a */proc* könyvtár alá fűztünk be. Az itt található állományok az éppen futó folyamatokról, illetve a rendszermag különböző környezeti változóinak állapotáról tartalmaznak értékes adatokat. Ezeket az állományokat egyik felhasználó sem írhatja, csupán olvashatja. A */proc*-ra nagy szükség van, ugyanis ha nincs befűzve, rengeteg szolgáltatás elérhetetlenné válik, sőt, olyan alapvető parancsokat sem használhatunk, mint a `ps`.
Ezek után nézzük meg, miként tudunk lemezegységeket kézzel be- és kifűzni! A befűzésre a `mount` parancs használható. Ha az adott lemezegység már szerepel a */etc/fstab* állományban, akkor beállításként elégséges csak a `mountpoint`-ot vagy az eszköz nevét megadnunk, például: `mount /mnt/cdrom`. Amennyiben mégsem írtuk bele, a `mountpoint`-on és az eszköznéven kívül a fájlrendszer típusát is meg kell mondanunk. Ezt a következő formában tehetjük meg: `mount-t fÆjlrendszer-t pus eszk znØv mountpoint`.
A lemezegységek kifűzését az `umount` utasítással végezhetjük. Itt kapcsolóként elég csak a `mountpoint`-ot megadnunk, ugyanis a */etc/mtab* nevű fájlban az összes idáig befűzött lemezegység adatai megtalálhatók.
Meg kell említenünk, hogy manapság már rengeteg segédprogram létezik a lemezegységek ki- és befűzésének könnyebbé tételére, illetve az *fstab* állomány szerkesztésére. Legtöbbjük grafikus, mindenki számára könnyedén kezelhető, és a „felhasználóbarátabb" terjesztések önműködően fel is telepítik őket. Tehát aki nagyon elveszettnek érzi magát a befűzés világában, annak számára létezik segítség. Például a KDE-hez is számos apró alkalmazás készült.
Most már csak néhány lemezkezeléssel kapcsolatos segédprogramról kell szót ejtenünk. Ilyen például az `fsck`, amely linuxos fájlrendszerünk ellenőrzésére, illetve javítására szolgál. Az `fsck` általában minden 20. rendszerindításkor az összes linuxos lemezterületen ellenőrzést végez.
A linuxos lemezterületek formázására is lehetőségünk nyílik. Szerencsére ez a művelet FAT-es társaihoz képest sokkal gyorsabb. Ext2 lemezterületet az `mkfs eszk znØv` utasítással formázhatunk.
Röviden ennyit a Linux alatti lemezkezelés rejtelmeiről. Sorozatunk következő részében ismét visszatérünk az általunk annyira kedvelt parancssorhoz, és kicsit magasabb szinten tekintjük át a héj- vagy parancs- (shell) műveleteket, továbbá alapszinten parancsfájlt írni is megtanulunk.

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

A Sendmail megerősítése

Mick megvizsgálja a Sendmail biztonsági hiányosságait, és felépít egy internetes leveleket kezelő SMTP-átjárót.

Igen, ó igen, a Sendmail. Rajonghatunk sokoldalúságáért és elterjedtségéért, vagy éppen utálhatjuk nagyságáért, bonyolultságáért és biztonsági hibáiért. Az is megeshet, hogy a levélkiszolgálók világában új fiúként egyszerűen csak adunk egy esélyt a Sendmailnek (végül is a Sendmail kétségtelenül a legnépszerűbb nyílt forráskódú programcsomag az Interneten).
Nos, ha szóba kerül a biztonság, a divatos szemlélettel ellentétben a Sendmailt nem kell teljesen leírni, és arra sincs szükség, hogy a ***sendmail.cf*** ősi írásmódját elsajátítsuk (de tényleg nem árt, ha a keményvonalas Sendmail-guruk ismerik). Ebben a hónapban ezeket és más Sendmail biztonsági kérdéseket fogunk boncolgatni, és a Sendmail hasznos m4 makróit felhasználva gyorsan felépítünk egy biztonságos, de működőképes Simple Mail Transport Protocol (SMTP, azaz egyszerű levéltovábbító szabvány) átjárót az internetes levelek kezeléséhez.

Miért (vagy miért ne) használjunk Sendmailt?

A Sendmail az egyik legősibb internetes programcsomag, amit még mindig széles körben használnak. Elsőként a BSD UNIX 4.1c változatában jelent meg (1983 áprilisában), és egészen napjainkig saját kategóriájának legkedveltebb alkalmazása maradt. Az üzenettovábbító ügynökprogramok (MTA-k) között a Sendmail az Internet fő igavonója, ami a leveleket a hálózatok között megfelelően és (a végfelhasználó számára) átlátszóan közvetíti. Csakhogy a Sendmail sem mentes a hátrányoktól. Kedvező jellemzői közé sorolható: a Sendmail hatalmas felhasználói közösséggel bír, ennek eredményeképpen igen könnyű mind kereskedelmi forgalomban beszerezhető, mind ingyenes támogatást találni hozzá, nem is beszélve a gazdag elektronikus és nyomtatott leírásokról. Kiforrottságának hála elég megbízható és kiszámítható.
Kedvezőtlen jellemzői közé tartozik: a Sendmail hosszú története során elég sok „cruft”-ot (régi kódot) gyűjtött össze, így aztán lassan biztonsági hibáiról és „elhízottságáról” is elhíresült. Természetesen mindkét vád vitatható. Az évek során számos jelentős sérülékenységére derült fény, ugyanakkor napvilágra kerülésük után igen gyorsan ki is javították őket. A terebélyesség vádpontját illetően tény, hogy a Sendmail kódbázisa sokkal nagyobb, mint más MTA-ké (mondjuk a Qmailé vagy a Postfixé), és memóriafoglalása is kétségtelenül méretesebb – ennek azonban legalább annyira a monolitikusság (egyetlen végrehajtható állomány teszi elérhetővé a legtöbb Sendmail-képességet) az oka, mint a felgyülemlett régi kódsorok. Ha jobban belegondolunk, a hosszú évek során a Sendmail forrását már oly sok programozó vizsgálta át tüzetesen, hogy nehezen képzelhető, hogy az elmúlt húsz évet túl sok kizárólag csak történelmi jelentőségű és elavult kód élte volna túl érintetlenül.
Sokkal hasznosabb a monolitikusság kérdését vizsgálni. A Sendmailnek bizonyos feladatok ellátásához néha rendszergazdai jogosultsággal kell futnia, például ha több különböző felhasználó saját könyvtárába ír leveleket. Emiatt aztán a Sendmail az olyan rendszereken, ahol kizárólag levéltovábbító (e-mail relay) vagy átjárófeladatokat lát el, csakis különleges jogosultságok nélküli (unprivileged) felhasználóként futhat.
A Sendmailt összetettségéért is bírálják. Beállításfájljának, a ***sendmail.cf***-nek szövevényessége nem éppen ösztönző, hogy mást ne mondjak – véleményem szerint valahol a C programnyelv és a szabványos kifejezések közti nehezen behatárolható helyen helyezkedik el. Mindezek oka természetesen a Sendmail különleges hatékonysága (bár sokan szeretnénk, ha a Sendmail inkább a C-t, a szabványos kifejezéseket vagy más, kicsit szabványosabb beállításnyelvet használna a ***sendmail.cf***-ben – legalább ennyire bonyolult, de egyedi saját nyelve helyett). Manapság már ez a pont is erősen vitatható. A Sendmail jelenlegi változatait már m4 makrókon keresztül állíthatjuk be, amelyek sokkal kevesebb felhasználóellenes élményt okoznak, mint a ***sendmail.cf*** kézi szerkesztése.
Egyes felhasználók véleményétől függetlenül a Sendmail megkérdőjelezhetetlenül hatékony és jól támogatott program. Ha a Sendmail előnyeit többre tartjuk a hátrányainál, akkor jó csapatban vagyunk. Azonban még ennél is jobb csapatba kerülhetünk, ha a Sendmail biztonságos futtatását is elsajátítjuk.

A Sendmail felépítése

Mint korábbiakban is említettük, a Sendmail monolitikus felépítésű a tekintetben, hogy minden tényleges munkát egyetlen végrehajtható állomány (maga a Sendmail) végez. A Sendmailnek két működési módja létezik: meghívható igény szerint, ebben az esetben feldolgozza a várakozó leveleket, majd kilép; avagy állandóan futó háttérdémonmódban is elindíthatjuk.
A démonmód csak akkor szükséges, ha a kívülről jövő levelek fogadása is a Sendmail feladatai közé tartozik; amennyiben viszont kizárólag levélküldésre használjuk, nem kell démonként futtatnunk, sőt, tulajdonképpen akár itt abba is hagyhatnánk az olvasást, hiszen a Sendmailnek ehhez semmi szüksége további beállításokra – hacsak nem akarjuk chrootolva futtatni.
Az, hogy a Sendmail miképpen működik, erősen attól függ, hogyan indítottuk el. Ha démonként (azaz a `-bd` kapcsolóval) futtatjuk, a 25-ös TCP-kapun figyeli a bejövő SMTP-kapcsolatokat, és időnként megkísérli elküldeni a ***/var/spool/mqueue*** nevű kimenősor könyvtárában összegyűlt leveleket. Ha csak úgy meghívtuk, akkor azt a kimenő levelet próbálja meg kézbesíteni, amiért meghívtuk, illetve a ***/var/spool/mqueue*** könyvtárat ellenőrzi egyéb, esetleg még várakozó kimenő leveleket keresve.

A feladat

Mielőtt továbblépnénk, szeretném egyértelműsíteni, mit is akarunk felépíteni. Példának az SMTP-átjárót választottam, mivel erre a feladatra egyrészt gyakran használják a Sendmailt, másrészt ennél a szerepnél igen sokat számít a biztonságosság (a legtöbb szervezetnél a nyilvánosság számára is elérhető levélkiszolgálók sokkal komolyabban fenyegetettségnek vannak kitéve, mint a belső levélkiszolgálók).
Az SMTP-átjárók esetében általában különös figyelmet kell

fordítani a jogosultsági szintekre, a fájljogosultságokra, és általában csak annyi szolgáltatást szabad engedélyezni, amennyi a levél célbajuttatásához valóban szükséges. Egy ilyen kiszolgálón a Sendmailnek lehetőleg jogosulatlan felhasználóként kell futnia; és csakis végső esetben – tehát amikor fájlokat kell írnia – szükséges chrootolni (a / egy alhalmazán), ugyanakkor úgy kell beállítanunk, hogy a leveleket csak a saját szervezetünknek továbbítsa, a levélszemétküldőkét (spammer) ne.
Red Hat 7 alatt nem túl sok trükk kívánatos a Sendmail SMTP-átjáró megerősítéséhez, illetve alig valamivel több lépés szükséges a SuSE vagy más terjesztések esetében.

A Sendmail beszerzése és telepítése

Teljes biztonsággal állíthatom, hogy az általunk kiválasztott Linux-terjesztés egy vagy több Sendmail-csomagot is tartalmaz. Természetesen az, hogy tényleg fel van-e telepítve a rendszerre, és hogy az általunk használni kívánt megfelelő változatról van-e szó, már más kérdés.
Ha rpm-alapú terjesztést használunk (Red Hat, Mandrake, SuSE stb.), a következő parancs kiadásával nézhetjük meg, hogy valamilyen változatú Sendmail fel van-e már telepítve:
`rpm -qv sendmail`
(a Debian-felhasználóknak a következőt kell beírniuk: `dpkg -s sendmail` – a fordító). A Red Hat és leszármazottai a Sendmailt három csomagra bontják: `sendmail`, `sendmail-cf` és `sendmail-doc`. A SuSE ellenben egyetlen `sendmail` nevű csomagot használ.
Tehát melyik változatot is futtatjuk? Ez idő tájt, amikor e sorokat írom, a legújabb Sendmail-változat a 8.12.2. A Red Hat 7 és a SuSE 7 viszont még mindig a Sendmail 8.11 különböző változatait támogatja. Amennyire tudom, semmi gondunk nem lesz, ha a terjesztésünk által támogatott Sendmail-változatnál maradunk, feltéve, hogy az legalább 8.11.0 vagy magasabb változatszámú. A 8.10-es és 8.11-es változatokban nem volt nagyobb biztonsági hiba; a 8.11, tulajdonképpen „bővített" kiadás volt: nem biztonsági lyukak foltozását tartalmazta, hanem azért adták ki, mert a Sendmail-csapat ekkor adta az SMTP-hez a TLS-titkosítást és az SMTP AUTH-továbbfejlesztéseket.
Természetesen amennyiben akad némi időnk és kedvünk, soha nem árt, ha a legfrissebb üzembiztos változatot fordítjuk és telepítjük. A józan ész kedvéért írásunk további részeiben feltételezem, hogy a Sendmail 8.10.0 vagy újabb változatát használjuk (az ettől eltérőt külön megemlítem).

Megjegyzés Debian-felhasználók számára

A Debian GNU/Linux v2.2 (Potato) még mindig a Sendmail v.8.9.3-at használja. Annak ellenére, hogy megbízható és viszonylag biztonságos kiadás, mostanra már két fő változattal is lemaradt (már amennyiben valaki, mint például én is, a második számot tekinti fő változatnak, hiszen az első szám már egy fél évtizede a nyolcas). Továbbá a 8.9.3 nem támogatja a TLS- és az SMTP AUTH-lehetőségeket (hamarosan, várhatóan május elején megérkezik a Woody Debian GNU/Linux v3.0 – a fordító).
Amennyiben TLS-t vagy SMTP AUTH-ot szeretnénk használni, vagy egyszerűen csak nem kívánunk régi változatot futtatni, még mindig eltávolíthatjuk a csomagot, letölthetjük a legfrissebb forráskódcsomagot a ➲ http://www.sendmail.org-ról, majd a Sendmailt forráskódból lefordíthatjuk és telepíthetjük. A forráskódcsomag jól leírt és Linux alatt könnyedén lefordul, feltételezve természetesen, hogy működőképes `gcc`-telepítéssel rendelkezünk.
Miután a Sendmailt akár bináris csomagként a terjesztésből, akár forráskódból fordítottuk és telepítettük, akad még pár feladat, amit nem árt elvégezni, mielőtt a Sendmail végrehajtható állományát démonként kezdenénk futtatni.

A SuSE Sendmail előkészítése

Ha SuSE-t használunk – amennyiben eddig még nem tettük volna meg –, váltsunk rendszergazdai jogosultságra. Nyissuk meg a */etc/rc.config* fájlt a kedvenc szövegszerkesztőnkkel, és az SMTP-változót állítsuk „yes"-re. Ez feltétlenül szükséges, ha azt szeretnénk, hogy a Sendmail */etc/init.d* könyvtárban található indítóparancsfájlja a rendszer indulásakor lefusson.
Továbbá a */etc/rc.config.d/sendmail.rc.config* fájlt át kell szerkesztenünk, hogy a `SENDMAIL_TYPE` változó „no"-ra legyen állítva. Ezáltal tulajdonképpen azt gátoltuk meg, hogy a *SuSEconfig* felhasználja a */etc/rc.config.d/sendmail.rc.config*-ot,

amely egyéb esetekben önműködően egy egyszerű Sendmail-beállítást hozna létre. Mi azonban most egy SMTP-átjárót, illetve továbbítórendszert szeretnénk beüzemelni, ami igencsak túlmutat a *sendmail.rc.config* képességein. Ha gépünk csak egyszerű SMTP-kiszolgálóként fog működni a saját helyi felhasználóihoz, valószínűleg ezt az egy fájlt elég átszerkesztenünk (ehhez előbb a `SENDMAIL_TYPE` változót „yes"-re kell állítanunk); ha ezt választanánk, a *sendmail.rc.config* teljes leírását megtaláljuk a */etc/mail/README* fájlban.
Miután az *rc.config* és a *sendmail.rc.config* állományokat átszerkesztettük, futtassuk le a *SuSEconfig*-ot. Ezzel érvényt szerezhetünk az *rc.config* és *sendmail.rc.config* fájlokban imént elvégzett változtatásoknak. A démon elindításához begépelhetjük a `/init.d/sendmail start` parancsot, azonban én azt javaslom, előbb inkább várjuk meg, amíg a Sendmailt teljesen beállítjuk.

A Red Hat Sendmail előkészítése

A Red Hat-felhasználóknak a Sendmail beállítása előtt mindössze egyetlen lépést kell elvégezniük: át kell szerkeszteni a */etc/sysconfig/sendmail* fájlt, hogy a `DAEMON` változó értéke „yes" legyen. Ez mondja meg ugyanis a */etc/init.d/sendmail* indítóparancsfájlnak, hogy rendszerindításkor a Sendmailt démonként kell futtatnia.

A Sendmail beállítása

Végre-valahára nekikezdhetünk a Sendmail beállításának tartományunk SMTP-átjárójaként. A következőkben leírtak a 8.9-es felett a Sendmail bármely változatára érvényesek (semmilyen körülmények közt ne futtassunk a 8.8-as változatot!).
A Sendmail beállításfájl (***sendmail.cf*** és a hozzá tartozó fájlok) egyszerűsített változatának előállításához a következő lépéseket kell megtennünk:

1. A ***sendmail.mc***-ben engedélyezzük a szükséges képességeket.
2. Ha szükséges, a ***sendmail.mc***-ben állítsuk be a tartománynév-álcázást (domain-name masquerading).
3. Futtassuk le az m4-et, amely a ***sendmail.mc***-ből létrehozza a ***generate.cf*** fájlt.
4. A ***mailertable*** szerkesztésével állítsuk be a kézbesítési (delivery) szabályokat.
5. Az ***access*** szerkesztésével állítsuk be a továbbítási (relay) szabályokat.
6. Az ***aliases***-ben állítsuk be a helyi felhasználói álneveket.
7. A ***mailertable***, ***access*** és ***aliases*** állományokat alakítsuk adatbázissá.
8. Az összes helyi gépnevet adjuk meg a ***local-host-names***-ben.
9. Indítsuk (újra) a Sendmailt.

A „sendmail.mc" érdekesebb beállításai

Az első és valószínűleg legidőigényesebb feladat az SMTP-átjáró felállítása során a ***/etc/sendmail.cf*** előállítása. Ezt legkönnyebben a ***/etc/mail/sendmail.mc*** átírásával tehetjük meg (SuSE-rendszereken e fájl neve ***/etc/mail/linux.mc*** – más terjesztések alatt eltérő is lehet).
A használt Linux-terjesztéstől függően a ***sendmail.mc*** beállítási adatait a ***/usr/share/doc/sendmail/README.cf*** (Red Hat és társai) a ***/usr/share/sendmail/README*** (SuSE) vagy valamilyen más állományban találjuk. Nincs elég hely arra, hogy e fájl számos beállítási lehetőségét részletekbe menően ismertessem. Megvizsgálom viszont azokat, amelyek biztonság tekintetében hasznosak lehetnek vagy beállításainkat modularizálják.
A Sendmail beállítását magán a ***sendmail.cf***-en kívül, külső fájlokból beolvasott adatokkal is megoldhatjuk. Ez két okból is célszerű: egyrészt a ***sendmail.cf*** közvetlen szerkesztése elég kényelmetlen, a ***sendmail.mc***-ből történő újralétrehozása pedig nem mindig kívánatos. Másrészt amennyiben SMTP-átjárónkon több különböző jogosultsággal rendelkező rendszergazda is lesz, jól jöhet, ha a ***sendmail.cf*** fájlt zárva tartjuk, ugyanakkor a többi rendszergazdának megengedjük az álnevek és a levéltovábbítási szabályok szerkesztését (vagyis a ***/etc/mail/access*** és ***/etc/mail/mailertable*** fájlok módosítását).
A leghasznosabb külső beállításfájlok, amelyeket érdemes engedélyezi:

- a ***mailertable***, amely a helyi kézbesítési szabályokat írja elő;
- a ***virtusertable***, ez virtuális tartománymegfeleltetéseket ír le felhasználónkénti és tartományonkénti bontásban;
- az ***access***, ami meghatározza, hogy mely gépek használhatják a kiszolgálót SMTP-továbbítóként.

A fenti fájlokat engedélyező ***sendmail.mc***-utasítások a következők:

```
FEATURE('mailertable',
➥'hash -o /etc/mail/mailertable.db·)dnl
FEATURE('virtusertable',
➥'hash -o /etc/mail/virtusertable.db·)dnl
FEATURE('access_db','hash -o
➥/etc/mail/access.db·)dnl
```

(A ***mailertable*** és `access_db` képességek Red Hat alatt alapértelmezetten érvényesek, viszont a ***virtusertable*** részt kézzel kell hozzáadni.)
Minden sor arra utasítja a Sendmailt, hogy az adott fájlra hivatkozást készítsen (bár az elérési adatbázist ***access***-nek neveztük `access_db` helyett), és annak hash-adatbázisát, illetve elérési útját használja. Hamarosan megismerhetjük, hogy hogyan használjuk fel ezeket a fájlokat, előbb viszont végre kell még néhány dolgot hajtanunk a ***sendmail.mc***-ben.
Ha felhasználóink elektronikus címei tartományunk szerintiek és a gép szerint, ahová bejelentkeztek, nem változnak – azaz ***mick@polkatistas.org*** formátumúak ***mick@myron.polkatistas.org*** formátum helyett, akkor kimenő leveleik ***From:*** mezőjét valószínűleg ennek megfelelően érdemes megváltoztatni (az ilyen általános címeken fogadott levelek felhasználói álneveket igényelnek – lásd később).
A következő sorok olyan ***sendmail.mc***-beállításokat mutatnak be, melyek arra utasítják példa-SMTP-átjárónkat, hogy a ***polkatistas.org*** felhasználóitól érkező levelek ***From:*** mezőjét az előbbieknek megfelelően írja át. A lenti példa összes sorát be kell szúrni, vagy a megjegyzésből ki kell szedni:

```
MASQUERADE_AS('polkatistas.org·)dnl
MASQUERADE_DOMAIN('.polkatistas.org·)dnl
EXPOSED_USER('root·)dnl
FEATURE('masquerade_entire_domain·)dnl
FEATURE('masquerade_envelope·)dnl
```

- A `MASQUERADE_AS` direktíva azt a teljes értékű tartománynevet adja meg, amit a megfelelő ***From:*** címekben szeretnénk látni.
- A `MASQUERADE_DOMAIN` direktíva adja meg azt a gépet, amire a `MASQUERADE_AS` vonatkozik. A ***polkatistas.org*** előtt álló "." azt jelenti, hogy az összes ebbe a tartományba tartozó gépnevet álcázni kell.
- Az `EXPOSED_USER` azt a felhasználónevet adja meg, amelynek ***From:*** címét nem szabad álcázni. A rendszergazda gyakori vendég e mezőben, mivel a tőle érkező levél sokszor figyelmeztetéseket és riasztásokat tartalmaz – ha ilyet kapunk, általában azt is tudni szeretnénk, hogy melyik géptől érkezett.
- A `masquerade_entire_domain` képesség azt jelenti, hogy a `MASQUERADE_DOMAIN` teljes tartományként és nem gépnévként értelmezendő; a `masquerade_envelope` eredményeképpen az álcázás nemcsak az SMTP-fejlécre vonatkozik, hanem a borítékra is.

Négy másik direktívát – egy logisztikait és három biztonsági jellegűt – találunk az *1. listában*.

- Az `always_add_domain` képesség Red Hat és SuSE alatt alapértelmezetten be van kapcsolva; az `use_cw_file` és `smrsh` Red Hat alatt érvényes, a SuSE alatt viszont nem; a `confSAFE_FILE_ENV` beállítást pedig minden esetben nekünk kell megadnunk.

1. lista Néhány további sendmail.mc-képesség

```
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
FEATURE('smrsh·,'/usr/sbin/smrsh·)dnl
define('confSAFE_FILE_ENV·,
'/var/mailjail·)dnl
```

2. lista A /var/mailjail tartalma

```
/var/mailjail:
total 1
drwx------   5 mail    mail    1024 Jan 22 17:09 var

/var/mailjail/var:
total 3
drwx------   4 mail    mail    1024 Jan 22 17:07 spool

/var/mailjail/var/spool:
total 2
drwx------   2 mail    mail    1024 Jan 22 17:06 mail
drwx------   2 mail    mail    1024 Jan 22 17:06 mqueue

/var/mailjail/var/spool/mail:
total 98
-rwx------   1 mail    mail    48528 Jan 22 17:06 bobo
-rwx------   1 mail    mail    47627 Jan 22 17:06 root

/var/mailjail/var/spool/mqueue:
total 0
```

- Az `always_add_domain` képesség a gép tartománynevét egyszerűen minden olyan levélhez kötelezően hozzáadja, amely magát tartománynév nélkül azonosító gépről érkezik. Például ha az SMTP-átjáró levelet kap ***bobo*** felhasználótól egy olyan gépről, amely magát csak ***whoopeejohn*** néven azonosította, a Sendmail a ***From:*** mezőt az eredeti ***bobo@whoopeejohn*** helyett ***bobo@whoopeejohn.polkatistas.org*** formátumúra írja át (de az álcázási direktívák itt is érvényesek).
- A `use_cw_file` képesség használata arra utasítja a Sendmailt, hogy a Sendmail által helyinek értékelt gépek listáját a */etc/mail/local-host-names* fájlból vegye. A */etc/mail/local-host-names* egyszerű szöveges állomány, amely soronként egyetlen gépnevet tartalmaz. Tegyük fel, hogy példa-SMTP-átjárónk nemcsak a ***polkatistas.org*** tartományból kap levelet, hanem a ***tubascoundrels.net***-ről is. Ha az átjáró neve `mail`, a ***local-host-names*** fájlja a következőképpen fog kinézni:

```
localhost.localdomain
mail.polkatistas.org
mail.tubascoundrels.net
```

Az 1. listában megadott harmadik képesség az `smrsh`, azaz a Sendmail korlátozott héjprogram. Ez nagyon fontos biztonsági lehetőség, amely képes korlátozni felhasználók ***.forward*** fájljából végrehajtható parancsokat.
Az 1. lista negyedik sora azt mondja meg a Sendmailnek, hogy a ***sendmail.cf*** `SafeFileEnvironment` változóját állítsa be – ahogy már biztosan ki is találták – a saját könyvtár (/) egy olyan alkönyvtárára, ahová a Sendmail chrootolni fog (mármint amelyik így lett beállítva). Jelenleg ez csak akkor következik be, amikor a Sendmail fájlokat ír. Ha meggondoljuk, ennek az ötvenszázaléknyi chrootolásnak is van értelme: pontosan a fájlírások azok, amelyek miatt a leginkább aggódnunk kell, és egy ilyen chrootkörnyezet kialakítása sokkal egyszerűbb, mintha a sok helyen használt `chroot jail`-t választanánk (abban az esetben ugyanis a chrootolt program által igényelt minden fájlszerkezetből, fájlból, végrehajtható állományból és eszközből másolatot kell tartani).
A 2. *lista* az én példa-`SafeFileEnvironment`-em */var/mailjail* könyvtárának teljes listáját (`ls -lR`) mutatja be. A */var/mailjail/var/spool/mqueue/bobo* és */.../root* fájlokat a Sendmail hozta létre. Ez előtt az egész `chroot jail`-környezetet mindössze négy paranccsal hoztam létre:

```
mkdir -p /var/mailjail/var/spool/mail/var/
➥mailjail/var/spool/mqueue
cd /var/mailjail
chown -R mail:mail *
chmod -R 700 *
```

Ha valakit a kéretlen kereskedelmi levelek témaköre érdekelne, számára is akad néhány jó hírem: a Sendmail alapértelmezés

Mick széljegyzete

Nem kell túl nagy jelentőséget tulajdonítani neki, de jómagam Postfix-rajongó vagyok. Postfixet és nem Sendmailt futtatok a saját tartományom SMTP-átjárójaként (igaz, saját hálózatomon helyi levéltovábbításhoz Sendmailt használok). Így aztán az e cikkben leírtakból, ideértve egyáltalán a létezését is, senki ne következtessen arra, hogy úgy gondolom, a Sendmail a legjobb választás, ha valakinek MTA-ra van szüksége – ezt mindenkinek magának kell eldöntenie.
Megkockáztatva, hogy kétértelműnek tűnök, azt kell mondanom, az elmúlt években meglehetősen sok időt fordítottam a Sendmailre és sokat segítettem másokat is a Sendmail használatában. Úgy vélem, hogy sokkal jobb, mint amennyire némelyek becsülik. Tapasztalataim szerint egyáltalán nem az a döcögő, cammogó, törékeny szörny, mint amilyennek néhány kritikusa beállítja. Valójában a Sendmailt igen megbízhatónak és hatékonynak tartom, mégha kicsit ijesztő is a bonyolultsága. Továbbá a legutóbbi, 1997-es CERT-tanácsadó óta (number CA-1997-05), amely a Sendmail biztonsági hibáját is tartalmazta, egyszerűen nem látom bizonyítottnak, hogy a Sendmail örökletesen ne lenne biztonságossá tehető. (A Sendmail átvizsgálása nyilván nem lett kevésbé szigorú az elmúlt öt évben, mint korábban.) Ezért azt hiszem, hogy bár más MTA-k (köztük a Postfix, Qmail és az Exim) egyértelmű előnyökkel rendelkeznek a Sendmaillel szemben a teljesítmény és a biztonság terén, egyben úgy vélem, a Sendmail elég jó minőségű ahhoz, hogy megkapja a Paranoid Pingvin minősítést (emellett az MTA-k „királyi családjából” származik: a beltenyészet miatt ugyan aggódhatok egy kicsit, de ettől még tisztelettel tartozom neki).

szerint nem engedélyezi az SMTP-továbbítást (relaying), ezt a levélszemétküldők által általánosan használt módszert. A szolgáltatást ugyan a ***sendmail.mc***-ben ki lehet kapcsolni, de tőlem ugyan meg nem tudod, hogyan. Inkább hagyjuk úgy. Továbbá a Sendmailt oly módon is beállíthatjuk, hogy minden ismert levélszemétforrásból érkező levelet utasítson el, amely a Realtime Blackhole List (RBL) feketelistáján szerepel. A következő sort kell csupán beszúrunk, illetve a megjegyzésből kiszednünk:

```
FEATURE('dnsbl')
```

Ahhoz azonban, hogy ez tényleg működjön, előbb fel kell iratkozni az RBL-re – honlapjuk címét megadtuk a ***Kapcsolódó címek*** között. A weblapon feliratkozási és felhasználási tanácsokat olvashatunk, illetve néhány fontos nyilatkozatot találunk (nem árt tudni, hogy míg a RBL-feliratkozás az egyéni, illetve hobbihelyek számára (Individual/Hobby Sites) ingyenes, ennek a szolgáltatásnak díjszabása is van). Az RBL felhasználásával a jogosult felhasználók leveleit éppúgy megállíthatjuk, mint a levélszemetelőkét, ezért kezeljük óvatosan.
Ha Red Hat 7.1-es vagy 7.2-es változatot használunk, létezik még egy ***sendmail.mc*** lehetőség, amit érdemes megnézni – ezúttal egy olyan, amit megjegyzésbe kell tenni. Amennyiben a ***/etc/mail/sendmail.mc*** fájlunk egy ilyen sort tartalmaz:

```
DAEMON_OPTIONS('Port=smtp,Addr=127.0.0.1,
➥Name=MTA·)
```

Tegyük megjegyzésbe, úgy, hogy a `dnl` karaktereket a sor elejére írjuk. Ameddig működik, ez a sor megakadályozza, hogy a Sendmail saját hurokeszközének (loopback) csatolófelületén kívül bármilyen más hálózatról érkező kapcsolatot fogadjon. Mondanunk sem kell, hogy egy SMTP-átjáró esetében ez nemkívánatos (bár kétségtelenül növeli a biztonságot). Ezek voltak a számunkra legfontosabb ***sendmail.mc***-beállítások. Léteznek természetesen más, biztonsági szempontból fontos beállítási lehetőségek is, különös tekintettel a nem átjárószabályokra (helyi kézbesítés stb.). További tájékoztatásért olvassuk el a ***README.cf*** vagy a README fájlokat, amelyeket e rész elején említettem.
Macro-beállítófájlunk ***sendmail.cf***-fé alakításához a következő parancsot használjuk:

```
m4 /etc/mail/sendmail.mc > /etc/sendmail.cf
```

Amennyiben macro-beállításfájlunk neve nem ***sendmail.mc***, helyettesítsük be a ***linux.mc*** vagy az éppen használt macro-beállítófájlnévvel. A Sendmail azt várja, hogy beállításfájlját ***sendmail.cf***-nek nevezzék; ráadásul mindig a ***/etc*** könyvtárban keresi, így aztán a parancs e része terjesztéstől, sőt, Sendmail-változattól függetlenül mindig így néz ki.

A kézbesítési szabályok beállítása

A nehezén túl vagyunk, most már csak azt kell a Sendmailnek megmondanunk, hogy mit csináljon a beérkezett levelekkel, milyen helyi gépnevek fogadhatók el, és milyen felhasználók, hálózatok és tartományok használhatják az SMTP-átjárót nem helyi célú levelek küldésére.
A ***mailertable***-t a kézbesítési szabályok megadására használjuk. Egyszerű az írásmódja, és terjesztéstől függően a ***/usr/share/doc/sendmail/README.cf*** vagy a ***/usr/share/sendmail/README*** fájlban találhatjuk meg. Dióhéjban: minden sor két részből áll, a célazonosítóból és a műveletből. A célazonosítónak kell megegyeznie a cél címével vagy annak egy részével; a művelet azt mondja meg, hogy a Sendmailnek mit kell tennie azokkal az üzenetekkel, amelyek célja megegyezett az azonosítóval. Ha az azonosító „.”-tal kezdődik, akkor minden levélforráscím, amely a pont után megadott részre végződik, találatnak számít. Ha nem, a „@” jelet követő összes karakternek meg kell egyeznie. A ***bobo@weird-al.polkatistas.org*** nem fog egyezni a ***polkatistas.org*** címmel, de egyezni fog a ***.polkatistas.org***-gal. A művelet ***ügynök:cselekmény*** alakú, ahol az ügynök lehet a levelező (a mailer, amit a ***sendmail.mc/linux.mc MAILER()*** pontjában állíthatunk be), vagy a beépített ***local*** ügynök, illetve ***error***. A ***local*** ügynök természetesen azt feltételezi, hogy a levelet valamilyen helyi felhasználónak küldjük, amit a kettőspont után adunk meg (ha a kettőspontot semmi sem követi, akkor magában az üzenetben megadott felhasználót fogja használni). Alább egy ***mailertable*** látható két különböző cselekménnyel:

```
polkatistas.org
➥smtp:internalmail.polkatistas.org
mail.polkatistas.org  local:postmaster
```

A kézbesítési szabályokon felül a Sendmailnek tudnia kell, hogy mely elektronikus levélcélokat kell a helyi (az SMTP-átjáró) gépnév rokonának tekintenie. Ezeket a ***/etc/mail/local-host-names*** fájlban adhatjuk meg, soronként egyet:

```
mail.polkatistas.org
weird-al.polkatistas.org
1.23.234.2
```

Végül azoknak a listáját kell megadnunk, akiknek a továbbítást a ***/etc/mail/access*** átszerkesztésével engedélyezzük. Az írásmód nagyon egyszerű: minden sor egy forrásnevet vagy -címet tartalmaz, ami után egy cselekmény áll (a részletekért ismét a ***README.cf***-et vagy az ennek megfelelő állományt nézzük át rendszerünkön). A cselekmény lehet `RELAY` (továbbít), `REJECT` (elutasít), `DISCARD` (elvet), `OK` vagy `ERROR` (hiba). A gyakorlatban ezek között a `RELAY` a leghasznosabb cselekmény, mivel alapesetben minden más továbbítás elutasításra kerül. A `REJECT` és a `DISCARD` cselekményeknek csak akkor van haszna, ha egy adott `RELAY`-szabály alól akarunk kivételeket megadni. Íme, egy egyszerű ***access*** fájl:

```
localhost.localdomain      RELAY
localhost     RELAY
127.0.0.1     RELAY
192.168       RELAY
```

Ugye, észrevettük a valódi gépnevek hiányát a fenti példában? Ebben a példában az SMTP-átjáró mindössze kimenő továbbítást enged; a bemenő levelek kizárólag helyi címekre jöhetnek, és a kimenő továbbításoknak is olyan gépekről kell érkezniük, amelyek IP-címe a `192.168` számokkal kezdődik (ami az Interneten nyilvánvalóan nem megadható címtartomány). Kedvelem ezt a módszert (az IP-cím használatot), hiszen így az IP-címálcázást tűzfalszabályaimmal megakadályozhatom, igaz, nem tudom meggátolni a hamis ***From:*** levélcímek átadását (természetesen a te igényeid mások is lehetnek):

```
access
local-host-names
mailertable
```

Kifinomultabb Sendmail biztonsági eljárások

Az SMTP AUTH (a Sendmail 8.10-es változatától fölfelé) már azonosítási lehetőséget hozott az SMTP-műveletek világába, azaz képes megállapítani, hogy engedélyezheti-e a továbbítást. Ez különösen akkor hasznos, amikor a rendszerek vagy a felhasználók nem futtatnak saját MTA-t, mégis szeretnének leveleket küldeni, azaz a kifelé menő leveleket egy központi átjárón át muszáj küldeni.

Ha olyan SMTP-kiszolgálót futtatunk, amely más tartományokból érkező leveleket is továbbít, nem árt, ha megismerkedünk ezzel a képességgel, mivel igen fontos védelem a kéretlen kereskedelmi levelek ellen, amelyek elkövetői jelentős részben az SMTP-továbbításokban bíznak.

Már csak egyetlen fájl maradt, amin finomítani lehetne: ez az ***aliases***. Ez a fájl tartalmazza a felhasználók elektronikus leveleinek álnévlistáját. Általában egy SMTP-átjárónak nincs szüksége túlságosan részletes ***alias***-adatbázisra; egész tartományok (vagy virtuális tartományok) levélcímeihez jobb, ha inkább a felhasználói adatbázist használjuk (ezt azonban hely hiányában sajnos nem áll módomban leírni). Szerencsére eléggé magától értetődő, így nyugodtan szerkesszük át, ha szükséges.

A tárgyalt négy fájlból három: a ***mailertable***, ***access*** és ***aliases*** állományok közvetlenül nem használhatók fel a Sendmailhez, először adatbázissá kell alakítanunk őket. A */etc/mail* könyvtár egy hasznos kis `Makefile`t tartalmaz e célra. Használatához egyszerűen csak váltsunk a */etc/mail* könyvtárba, és gépeljük be a következő parancsot:

```
Make access.db mailertable.db
```

A fenti parancs az ***aliases*** fájlhoz nem lesz jó, mivel ennek saját eszköze van: a `newaliases`. Futtassuk le minden kapcsoló nélkül a `newaliases`-t, és megváltoztatott */etc/aliases* fájlunk önműködően */etc/aliases.db* fájllá alakul.

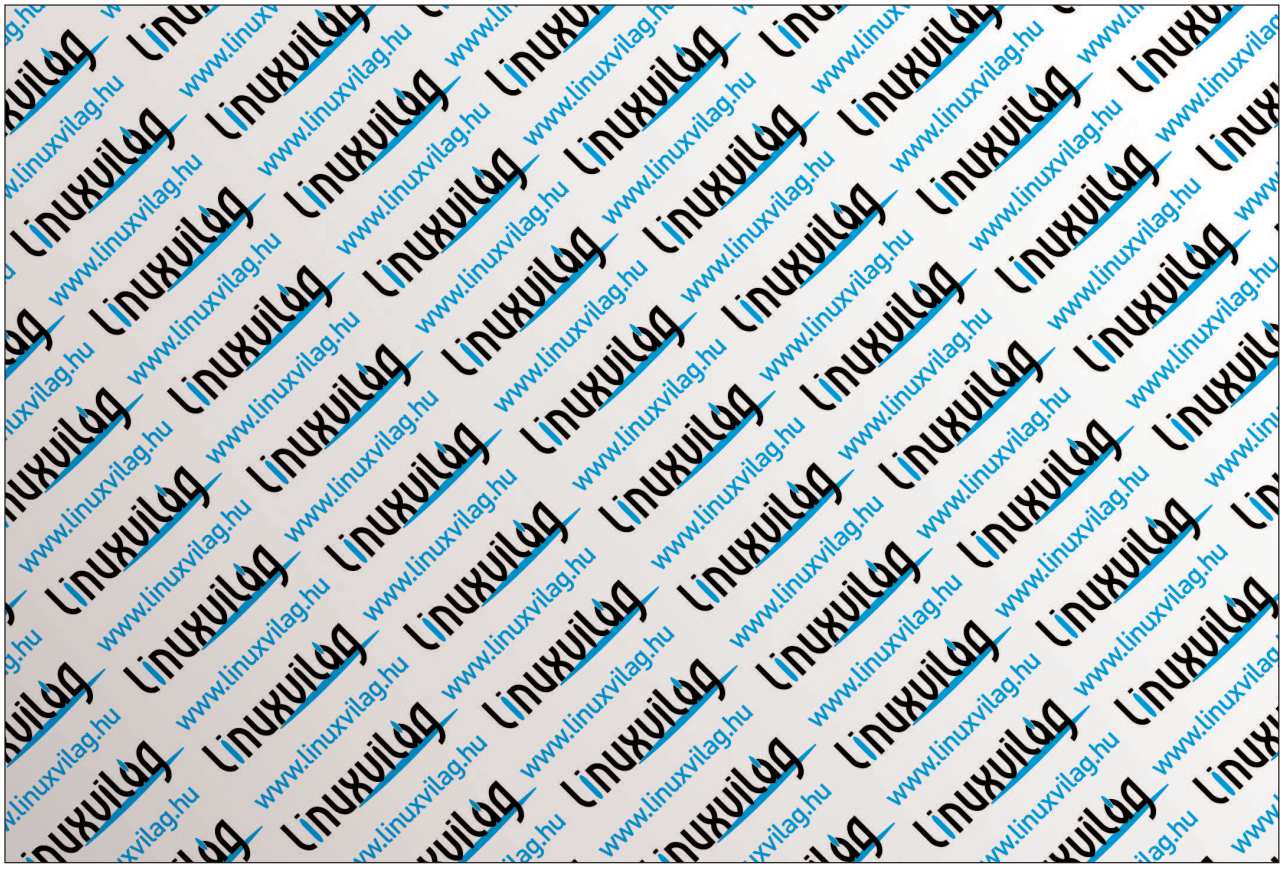
Egyelőre ennyi. Sok mindent nem sikerült elmondanom: külön kiemelném közülük az `smrsh` héjprogramot (amit főként a helyi levélkézbesítéshez lehet felhasználni és nem az átjárókhoz). Remélem, hogy azért sikerült néhány hasznos tippet adnom és útmutatást szolgáltatnom néhány teljesebb információforráshoz. Sok szerencsét!

Linux Journal március, 95. szám

Mick Bauer (mick@visi.com) hálózati biztonsággal foglalkozó szaktanácsadó. 1995 óta a Linux elkötelezett híve, 1997 óta pedig OpenBSD prófétaként tevékenykedik. Mick minden kérdést és megjegyzést szívesen fogad.

Kapcsolódó címek

- http://www.sendmail.net/000705securitygeneral.shtml
- http://www.sendmail.net/000710securitytaxonomy.shtml
- http:// www.itworld.com/Net/3314/swol-0699-security
- http://www.sendmail.net/810usingantispam.shtml
- http://www.sendmail.net/usingsmtpauth.shtml



A PPPD beállítása Linux alatt (2. rész)

Az első részben Tony a modem beállítási lehetőségeit mutatta be – most továbblép, és az Internet beállításával ismertet meg bennünket.

Írásunk előző részében (lásd a Linuxvilág 2002. májusi számát) felvázoltuk, hogyan is kell beállítani a modemet. A cikk végére érve egy */dev/modem* nevű közvetett hivatkozást kaptunk, amely a modem eszközfájljára mutatott. Megbizonyosodtunk arról is, hogy minden helyesen működik, vagyis összekapcsolódtunk a szolgáltatóval (igaz, PPP-kapcsolatot nem hoztunk létre). Ebben a cikkben továbblépünk, és megtudhatjuk, miképpen kapcsolódhatunk az Internethez. Javaslom, minden érdeklődő először az első cikket olvassa végig. Amennyiben ez lehetetlennek bizonyulna, legalább bizonyosodjunk meg afelől, hogy modem helyesen van-e beállítva, illetve hogy létezik-e a */dev* könyvtár megfelelő eszközére mutató */dev/modem*.
Hogy írásunknak hasznát vehessük, a szolgáltatónkhoz tartozó összes bejelentkezési adatra szükségünk lesz, ideértve a tárcsázandó telefonszámot, a belépési nevet és jelszót, illetve egy érvényes DNS-kiszolgáló címét (bár ez ugyan elhagyható, mivel a szolgáltató segítségével önműködően lekérdezhető). Cikkünk feltételezi, hogy szolgáltatónk elfogadja a PAP azonosítási protokollt. A PAP egy olyan módszer, amellyel azonosítónk és jelszavunk a PPP-protokollon keresztül a szolgáltatóhoz juttatható; ezáltal a felhasználó megtakaríthatja azt a bonyolultabb (gyakran kézi vezérlést igénylő) bejelentkezési folyamatot, amilyet a szolgáltatók rendszerei régebben igényeltek. Az ISP-k (internetszolgáltatók) döntő többsége manapság már megköveteli a PAP használatát. Írásunk azt is feltételezi, hogy hagyományos modemmel és nem úgynevezett Winmodemmel rendelkezünk. A Winmodem beállítása is megoldható, de fárasztó és tárgyalása meghaladja e cikk kereteit.

A kapcsolat kiépítése: alapeszközök

Mindenekelőtt az internetkapcsolat beállításához rendszergazdaként kell bejelentkeznünk. A modemen keresztüli internetkapcsolat a PPP (point-to-point, azaz két pont közötti protokoll) segítségével valósítható meg. A normál TCP/IP-csomagokat PPP-csomagokba zárjuk (encapsulate), így azok a soros vonalon keresztülküldhetők (ugyanis arról van szó, hogy a TCP/IP-csomagok hálózati átvitelhez készültek, és átalakítás nélkül nem férnek el a soros kapcsolaton).
Írásunkban feltételezzük továbbá, hogy gépünkön a következő programok megfelelően telepítve vannak:

- a PPP-kezelő rendszermagmodulok (minden általam ismert terjesztés rendszermagváltozata tartalmazza a PPP-modult, így emiatt valószínűleg nem kell aggódnunk);
- a PPPD program, amely a kapcsolat elindítása után alaphelyzetbe állítja a rendszermagmodulokat;
- a chatprogram, amely a kapcsolat létrehozásáért felelős;
- a minicom program, amely egy igen egyszerű terminálprogram, s amit a modemmel való kapcsolattartáshoz fogunk használni.

A programok meglétének ellenőrzésére a `which` parancsot használhatjuk. Ez a parancs ugyanis értesít bennünket, ha a keresett programfájl `$PATH` környezeti változóban felsorolt könyvtárak valamelyikében megtalálható:

```
which pppd
/usr/bin/pppd
which chat
/usr/bin/chat
which minicom
/usr/bin/minicom
```

Amennyiben nem rendelkeznénk valamelyik programmal, meg kell szereznünk a megfelelő csomagot, és fel kell telepítenünk.

Kapcsolatáttekintés

Miután beállítottuk a modemet, hogyan csatlakozhatunk az Internethez? A PPPD (point-to-point protokoll démon) nevű programot kell használnunk (és beállítanunk). A PPPD futtatásakor a következők történnek (feltételezve, hogy tárcsázóprogramként a `chat`-et használjuk, és a PPPD helyesen van beállítva). A démon elindul, beállítja a soros kapu jellemzőit (sebesség stb.). Ezt követően a kapcsolat létrehozásához egy külső programot futtat le (a `chat`-et), amely a modemhez elküldi a kapcsolatteremtő utasítást (`ATDT` parancs, amelyet a szolgáltató telefonszáma követ). Ezután a `CONNECT` karaktersorozat megérkeztére vár a soros kapun. Ha idáig eljutottunk, a kapcsolat már felépült, és úgy is vehetjük, mintha számítógépünket a szolgáltató gépével egy soros kábel kötné össze. Mikor a `chat` befejezte futását, a vezérlést ismét a PPPD veszi át. Ha a kapcsolatot nem lehet kiépíteni, a PPPD kilép és hibát ad vissza, egyébként elkezd „beszélgetni” a vonal másik végén üldögélő PPP démonnal (ez az a PPP-kézfogás, amit korábban egy halom érthetetlen karakter formájában láthattunk), és végül kap egy IP-címet (abban az esetben, ha dinamikus IP-címmel rendelkezünk – a ford.). Általában ehhez a lépéshez szükség van azonosítónkra és a jelszavunkra (a bejelentkezési adat a PPP-kézfogás alatt küldődik el). A PPPD program ellenőrzi a rendszermag hálózati csatolófelületének létrejöttét és hogy a hálózati forgalom ide irányítódjon.

Néhány szó a bejelentkezésről

A két szükséges program – a PPPD és a `chat` – nem interaktív. Egyszerűen csak lefutnak, és minden üzenetet a Syslog rendszernapló démonnak küldenek el (`system log daemon`). A `syslogd` azután a megkapott üzeneteket a merevlemezre írja. Az üzeneteknek több típusa is van, és a különböző típusok más-más fájlokban tárolódnak. A tárolás pontos helyét a `syslogd` beállításai határozzák meg. Ezért nem árt, ha beállítjuk a `syslogd`-t, így száz százalékig biztosak lehetünk benne, hogy a PPPD és chat démonoktól származó üzenetek valóban a merevlemezre kerülnek, illetve megtudjuk, hogy egyáltalán hova is kerülnek. A `syslogd` beállításfájlja a */etc/syslog.conf*. Mindössze egyetlen kiegészítő

sort kell beírnunk. Ehhez a következő parancsot gépeljük be:

```
vi /etc/syslog.conf
```

Természetesen tetszés szerinti szerkesztőt használhatunk (vi, Emacs, joe, pico stb.). Szúrjuk be a következő sort:

```
daemon.debug;*.info        /var/log/ppp_article
```

Ne feledjük, hogy az `info` és a `/var/log/ppp_article` között egy TAB-nak kell állnia.
Ezután el kell érnünk, hogy a `syslogd` démon észrevegye a beállításfájljában történt változtatást. Futtassuk a következő parancsot:

```
killall -HUP syslogd
```

Létre kell jönnie a */var/log/ppp_article* fájlnak, a belsejében pedig egyetlen sornak kell állnia, amely a `syslogd` újraindítását jelzi. Ellenőrzéséhez a következő parancsot írjuk be:

```
cat /var/log/ppp_article
Aug  4 19:28:46 merc_linux syslogd 1.3-3:
➥restart.
```

A `cat` parancs helyett, amely a fájlt csak olvassa, a `-f` kapcsolóval kiegészített `tail`-t is használhatjuk. Ez ugyanis folyamatosan olvassa a fájlt, és az újonnan bekerülő adatokat a képernyőre írja. Ez azt jelenti, hogy amint a `syslogd`, valamit a *ppp_article* fájlba ír, a `tail` azonnal megjeleníti a képernyőn:

```
tail -f /var/log/ppp_article
Aug  4 19:28:46 merc_linux syslogd 1.3-3:
➥restart.
```

Ettől kezdve a `chat` vagy a PPPD által rögzített adat önműködően megjelenik a képernyőn is. Erősen ajánlott, hogy ezt a konzolt folyamatosan nyitva tartsuk, és ellenőrizzük a rajta megjelenő üzeneteket, amikor csak szükséges.

Ismerjük meg a chatet!

Amint a cikk első részében kiderült, a soros kapcsolat kiépítéséhez az `ATDT12345678` (természetesen szolgáltatónk telefonszámával) karaktereket kell a modemhez eljuttatni, majd a modemtől a `CONNECT` karaktersorozatot várjuk (ami a kapcsolat létrejöttekor fog megérkezni). A `CONNECT`-en kívüli egyéb üzeneteket vissza kell adni: `BUSY`, `NO CARRIER`, `NO ANSWER` stb. Az előző cikkben a `minicom` segítségével ezt a gyakorlatban is kipróbáltuk.
Igaz, hogy ezeket a `minicom` segítségével kézzel is végre tudjuk hajtani, de nem árt, ha van egy programunk, ami megteszi helyettünk. A programnak tudnia kell beszélni a modemmel, képesnek kell lennie adatot küldeni és egy adott karaktersorozatra várakozni. Természetesen létezik ilyen program, a neve `chat`. Próbáljuk meg például a következő parancsot lefuttatni:

```
chat ABORT "BUSY" "OK" "TRY" "THIS" "TESTING"
➥"COMMAND"
```

Legyünk óvatosak, ugyanis mostantól a billentyűzet le van zárva, és nem lehet csak úgy kilépni a programból, még a CTRL-C leütésével sem. Gépeljük be az `OK` szót, amire a `TRY` üzenet jelenik meg. Most gépeljük be, hogy `THIS`, ezután a `TESTING` szó jelenik meg a képernyőn. Végül üssük be a `COMMAND` szót, amire a program sikeresen kilép. Próbáljuk meg újra lefuttatni a parancsot: írjunk `OK`-t, amire ismét megjelenik a `TRY` szó. Ezután üssünk `BUSY`-t, és a program azonnal kilép. Ahogy már kitalálhattuk, a `chat` programot arra tervezték, hogy karaktersorozatokat várjon, és valami mást írjon ki válaszul. A két első szó – `ABORT BUSY` – különleges jelentésű, és arra utasítja a `chat`-et, hogy lépjen ki, ha a végrehajtás során bármikor a `BUSY` szóval találkozik. Ha valami nem működik, a `chat` parancsot a `-v` kapcsolóval is lefuttathatjuk:

```
chat -v ABORT "BUSY" "OK" "TRY" "THIS"
➥"TESTING" "COMMAND"
```

A `-v` kapcsoló a `chat`-et beszédesre állítja, így mindig elmondja nekünk, hogy éppen pontosan mit is csinál vagy mit vár és így tovább. Természetesen minden hibakereső üzenet a */var/log/ppp_article*-re fog kerülni, ha követtük a `syslogd`-nál korábban leírt utasításokat.
Vizsgáljunk meg egy másik `chat` parancsot:

```
chat ABORT "BUSY" "" "AT" "OK" "ATDT93355100"
➥"CONNECT"
```

Ahogy valószínűleg mindenki kitalálta, úgy kell utasításokat kiadnunk, mintha mi lennénk a modem – amennyiben azt szeretnénk, hogy a `chat` sikeresen kilépjen. Egy `AT` karaktersorozatot fog küldeni nekünk, amire `OK` begépelésével kell válaszolnunk. Erre válaszul az `ATDT93355100` karaktersorozatot küldi nekünk, majd várakozik, amíg be nem gépeljük a `CONNECT` szót. Ezután kilép. Ez feltehetően a legtöbb olvasó számára ismerősen hangzik, pontosan ez kell nekünk az ISP-csatlakozáshoz, csak rá kell vennünk a `chat`-et, hogy a billentyűzet helyett a modemhez beszéljen. Én szolgáltatómhoz a következő parancsot használom:

```
chat ABORT BUSY ABORT "NO CARRIER"
➥TIMEOUT 120 "" AT OK ATDT94310999 CONNECT
```

Ez elég egyszerű, és igazság szerint sokkal jobban is lehetne csinálni, de nekem megfelel és elégedett vagyok vele. Megtekinthetjük a `chat` súgó oldalát is (egyszerűen gépeljük be: `man chat`), és elolvashatjuk a felajánlott lehetőségeket; később esetleg megváltoztathatjuk a kapcsolatteremtő parancsfájlt, hogy kihasználja a `chat` által felajánlott csinos kis megoldásokat. Következő lépésként egy parancsfájlt kell készítenünk, amely tartalmazza az imént megírt `chat` parancsot. A fájlt a */etc/ppp* könyvtárba helyezzük, és a *chat-connect* nevet adjuk neki. Létrehozásához gépeljük be a következő parancsot:

```
vi /etc/ppp/chat-connect
```

(természetesen bármilyen más szerkesztőt is használhatunk, ha a `vi`-t nem kedveljük). A parancsfájlnak a következőképpen kell kinéznie:

```
#!/bin/sh
chat ABORT BUSY ABORT "NO CARRIER"
➥TIMEOUT 120 "" AT OK ATDT94310999 CONNECT
```

A `94310999` helyére a saját ISP-szolgáltatónk számát kell írnunk. Ezután mentsük, és lépjünk ki a szerkesztőből.

A parancsfájlt futtathatóvá is kell tennünk, ezt érhetjük el a `chmod` paranccsal:

```
chmod +x /etc/ppp/chat-connect
```

Győződjünk meg parancsfájlunk működőképességéről a következő parancs futtatásával:

```
/etc/ppp/chat-connect
```

Ha működik, ismét egy lépéssel közelebb kerültünk az internetkapcsolathoz. Tulajdonképpen kifejezetten közel vagyunk a célhoz. Mindössze arra van szükség, hogy a PPPD-t a megfelelő kapcsolókkal futtassuk.

Ismerjük meg a PPPD-t!

Most már elkezdhetünk magával a PPPD beállításaival foglalkozni. Az ide vonatkozó fájlok: */etc/ppp/options*, */etc/ppp/chap-secrets*, */etc/ppp/pap-secrets* és */etc/ppp/peers*. Az *options* fájl a PPPD alapértelmezett beállításait sorolja fel. Egyelőre induljunk el úgy, hogy a */etc/ppp* könyvtárban található *options* fájl teljesen üres legyen; kedvenc szerkesztőnkkel mindent töröljünk ki belőle. Ha ezt nem szeretnénk megtenni, a tartalmat megjegyzéssé is változtathatjuk, amennyiben a sorok elé beszúrjuk a # jelet. Nagyon fontos, hogy üres *options* fájllal indítsunk, mivel csak így lehetünk benne biztosak, hogy tiszta lappal indulunk. Első lépésként nézzük meg, hogy az imént elkészített `chat` parancsfájlunk valós helyzetben is jól működik-e. Ehhez a PPPD-t kell elindítanunk néhány kapcsolóval:

```
pppd /dev/modem 38400 modem lock connect
➥/etc/ppp/chat-connect
```

A kapcsolókat a PPPD-nek tetszőleges sorrendben megadhatjuk. A `/dev/modem` kapcsoló azt a soros kaput jelképezi, ahová a modemet kötöttük (mint tudjuk, ez közvetett hivatkozás a valódi `ttyS`-eszközre). A `modem` kapcsoló arra figyelmezteti a PPPD démont, hogy modemes kapcsolaton keresztül fog működni, nem pedig egy egyszerű, minket és a szolgáltatót összekötő soros kábelen. A `word lock` azt jelenti, hogy a modemet – amíg használjuk – le kell zárni (ha valaki nem tudja, hogy ez mit jelent, ne aggódjon; csak arról van szó, hogy így garantálható, hogy amíg a kapcsolat él, más program ne férhessen a modemhez). Az utolsó lehetőség a `connect`, amely `/etc/ppp/chat-connect` kapcsolóval együtt értelmezendő, és azt mondja meg a PPPD-nek, hogy melyik programot kell elindítania a szám tárcsázásához és az internetszolgáltatóhoz való kapcsolódáshoz – a mi esetünkben ez most a cikk előző fejezetében megírt `chat` parancsfájl lesz.
Ha a dolgok nem működnének, `chat` parancsfájlunkat kiegészíthetjük `-v` kapcsolóval, majd újból próbálkozva figyeljük a naplófájlokat – ezen a ponton még viszonylag könnyen megoldhatjuk a nehézségeket. Ha minden jól megy, láthatjuk, amint modemünk csatlakozik, és hallhatjuk, amint a szokásos sípolási szertartás lezajlik. Most már csak egyetlen lépés választ el bennünket az internetkapcsolattól. Szerkesszük át a */etc/pap-secrets* fájlt, és jelszavunkat az alábbihoz hasonló sorral adjuk hozzá:

```
felhasznÆl i_neved   *      jelszavad
```

Ne feledjük, minden szó között egy TAB-nak kell lennie. Máris készen állunk a nagy próbára, a valódi kapcsolatfelvételre. Próbáljuk ki a következő parancsot:

```
pppd /dev/modem 38400 modem lock connect
➥/etc/ppp/chat-connect user
➥felhasznÆl i_neved defaultroute
```

Az egyetlen új kapcsoló a `user` (amit a */etc/ppp/pap-secrets* fájlban is szereplő felhasználói nevünk követ), illetve a `defaultroute` lehetőség. Ez utóbbi teszi lehetővé, hogy az Internetre igyekvő csomagok alapértelmezés szerint ezt a kapcsolatunkat használják. Ezzel a lehetőséggel a kapcsolat létrehozása után a PPPD be fogja állítani a megfelelő `routing`-tábla bejegyzéseket. A naplófájlban ilyesféle üzenetet kell látnunk:

```
Aug  4 16:12:23 merc_linux pppd[4430]:
➥local  IP address 94.232.195.174
Aug  4 16:12:23 merc_linux pppd[4430]:
➥remote IP address 194.232.195.4*
```

Amennyiben nem így történt, a `debug` kapcsolóval lefuttathatjuk a PPPD-t, és megnézhetjük a naplófájlt (azaz a */var/log/ppp_article*-t), hogy megvizsgáljuk, mi is történt:

```
pppd /dev/modem 38400 modem lock connect
➥/etc/ppp/chat-connect user
➥felhasznÆl i_neved defaultroute debug
```

Ha minden működött, gratulálok, immár kapcsolatban állsz az Internettel. Ne feledjük, a lecsatlakozáshoz csak ennyit kell beírnunk:

```
killall pppd
```

A kapcsolat kipróbálása

A következő lépés, hogy megnézzük, kapcsolatunk valóban működik-e. Ha meg akarjuk tudni, hogy él-e a kapcsolat vagy sem, a legegyszerűbb, ha az `ifconfig` parancsot futtatjuk le (lásd az 1. listát a 34. CD Magazin/PPPD könyvtárban). Ez a parancs a rendszermag jelenleg működő hálózati csatolóit mutatja. Nálam például létezik egy `lo`, azaz normál hurokeszköz (loopback) csatolófelület, amit akkor használok, ha magamhoz akarok csatlakozni, és egy `ppp0`, ami a modem PPP-csatolófelülete.
Ha látni kívánjuk, hogy tényleg kitalálunk-e az Internetre (működik-e a `routing`), futtassuk le a `traceroute` parancsot, amit valamilyen IP-cím követ. Egyelőre használjuk a `-n` kapcsolót, hogy a DNS-névfeloldást (name resolution) kikapcsoljuk (ezt ugyanis még nem állítottuk be).
Például:

```
traceroute -n 198.182.196.56
traceroute to 198.182.196.56 (198.182.196.56),
30 hops max, 38 byte packets

 1  194.232.195.4 (194.232.195.4) 181.518 ms
    139.473 ms  149.822 ms
 2  194.232.195.1 (194.232.195.1) 129.540 ms
    139.739 ms  139.821 ms
...
19  207.245.34.122 (207.245.34.122)  479.696 ms
    479.653 ms *
20  198.182.196.56 (198.182.196.56) 489.711 ms
    479.644 ms  479.874 ms
```

A `198.182.196.56` IP a ➲ http://www.linux.org névhez tartozik. A `traceroute` mutatja meg nekünk azt az utat, amin

az általunk küldött csomagok az Interneten utaznak. Itt az ideje, hogy a rendszernek a */etc/resolv.conf* fájl segítségével megadjuk DNS-ünk IP-címét. Az én *resolv.conf* állományom a következőképpen néz ki:

```
nameserver 203.14.168.3
nameserver 202.0.185.226
```

Néhány ISP nem ad DNS-kiszolgálócímeket, mivel a számítógép a PPP-kézfogás befejezésekor önműködően kap egyet. Ha ez a helyzet, egyszerűen lépjünk ki, majd a PPPD futtatásával a `usepeerdns` kapcsolóval kiegészítve indítsuk újra a kapcsolatot:

```
pppd /dev/modem 38400 modem lock connect
➥/etc/ppp/chat-connect user
➥felhasznÆl i_neved defaultroute usepeerdns
```

Rögtön meg is nézhetjük, hogy működik-e a DNS, például a Telnet programmal. A Telnetet most csak arra használjuk, hogy lássuk, képes-e a rendszer a www.linux.org nevet IP-címmé fordítani.

```
telnet www.linux.org 80
Trying 198.182.196.56...
Connected to www.linux.org.
Escape character is ·^]·.
```

Működik! Már indíthatjuk is a böngészőnket (Netscape, Mozilla, Opera, Galeon, Lynx stb.), és kedvünkre nézegethetünk a Hálózaton.

Egy kis utómunka

Végre minden jól működik: az internetkapcsolat él, és amikor csak akarunk, csatlakozni tudunk az Internetre. Akad azonban még lehetőség egy kis további fejlesztetésre. Az első dolog, amit érdemes megcsinálni, a soros kapcsolat sebességének növelése, majd ellenőrzése. Ehhez a PPPD parancssorban a `38400`-as számot `11 5200`-ra kell cserélnünk.
Néhány hét után valószínűleg fel fog tűnni, hogy rengeteg kapcsolót kell megadnunk a PPPD parancshoz. Tulajdonképpen minden egyes kapcsolathoz az egész sort be kell gépelnünk:

```
pppd /dev/modem 115200 modem lock connect
➥/etc/ppp/chat-connect user
➥your_username_here defaultroute
```

Jó hír, hogy ezeket a kapcsolókat természetesen be tudjuk rakni egy beállításfájlba, mégpedig a */etc/ppp/options* állományba. Így esetünkben az *options* fájl a következőképpen fog kinézni:

```
/dev/modem
115200
modem
lock
connect /etc/ppp/chat-connect
user your_username_here
defaultroute
```

Ebben a fájlban a kapcsolók sorrendje nem igazán számít. Ettől kezdve egyszerűen a `pppd` parancs begépelésével csatlakozhatunk az Internethez. Mi történik, ha több szolgáltatóhoz is szeretnénk csatlakozni? Ebben az esetben több *options* fájlt kell készítenünk, amelyeket aztán a */etc/ppp/peers* könyvtárba helyezünk. Az alábbi kimeneten látható, hogy néz ki az én *peers* könyvtáram:

```
ls -l /etc/ppp/peers
total 4
-rw-r--r--   1 root   root   197 Aug   4 15:41
➥main_net
-rw-r--r--   1 root   root   189 Mar 11   2000
➥primus
```

A */etc/ppp/options* állományom üres; amikor a PPPD-t futtatom, mindig ezt írom:

```
pppd call main_net
```

Ilyenkor a */etc/ppp/peers/main_net* és a */etc/ppp/options* fájljaim is értelmezve lesznek (melyek közül a második történetesen üres). Ha valamilyen okból kifolyólag a fő szolgáltatóm (Main Net) leállna, még mindig használhatom Primus időkorlátos fiókomat.
A legjobb, amit ezen a ponton tehetünk, hogy elolvassuk a PPPD kézikönyvoldalait (egyszerűen üssük be: `man pppd`) és megnézzük, hogy a sok kapcsoló közt akad-e olyan, ami esetleg a kapcsolatunknak hasznára lehetne. A 2. listában (34. CD Magazin/PPPD könyvtár) egy igen gazdag *options* fájlt találunk, amit *Pancrazio De Mauro* barátom, egy igazi Linux-guru írt. Vajon, te jobban meg tudod csinálni?

Összegzés

A folyamat talán kicsit félelmetesnek tűnhet; a Linux alatti internetkapcsolat kiépítéséhez szükséges tudásszint elképesztő, különösen a Windows *Remote access interface* egyszerűségével összehasonlítva; ennek fényében kérdéses, hogy megéri-e egyáltalán mindent kézzel beállítani. (Azért tegyük hozzá, hogy a legembertelenebbnek nevezett Debian alatt egy kezdőnek is mindössze fél percébe kerül belőni a PPP-t is – a `pppconfig` programmal. Szóval, aki korbáccsal veri magát, az ne sírjon, ha fáj – a fordító).
Véleményem szerint két nagy előnye van annak, ha mindent kézzel állítunk be: az egyik, hogy végig tudunk (sőt végig kell) menni sok PPPD-kapcsolón, ami kapcsolatunkat hatékonyabbá teheti. A második, hogy mostantól, ha az internetelérést grafikus felületen állítjuk be, tudni fogjuk, mi is történik, és sikerrel próbálkozhatunk a helyrehozással, ha az önműködő megoldással nem jutunk dűlőre.
Mielőtt befejeznénk, szeretnék rámutatni, hogy létezik egy parancssoros program (nem GUI), amelyik önműködően végrehajtja a cikkben leírt lépéseket, (megkeresi a modemet, a megfelelő kapcsolók használatával összekapcsolódik a szolgáltatóval stb.). A programot wvdial néven találhatjuk meg a ➲ http://www.worldvisions.ca/wvdial/index.html címen.

Linux Journal március, 95. szám

Tony Mobily
(merc@mobily.com) a Login olasz számítástechnikai magazin szakmai szerkesztője. LCI (Linux Certification Instructor, ➲ http://www.linuxcertification.com) képesítéssel rendelkezik, angol, olasz, C, Perl és néhány más nyelvet is használ.

Linux IPv6 – Melyik változatot telepítsem?

Ha még habozik, hogy melyik IPv6-megvalósítást válassza linuxos kiszolgálógépe számára, cikkünk segít a döntésben.

Az IPv6 – vagyis az Internet Protocol Version 6 szavakból alkotott elnevezés – következő nemzedékbeli protokoll, mely az Internet Engineering Task Force (az Internet felépítését felügyelő szervezet, a továbbiakban: IETF) által tervezett megoldás, a jelenlegi 4-es változat (IPv4) felváltására készült. Napjainkban az Internet még többnyire az IPv4-et használja, amely közel húszesztendős. Az IPv4 „élemedett" kora ellenére mindeddig figyelemreméltóan rugalmas volt, de mostanában gondok merültek fel körülötte. A legfontosabb ilyen az IPv4-es címek egyre fokozódó hiánya az Internetre kapcsolódó új gépek és készülékek körében.

Az IPv6 gyógyírt ad számos, az IPv4-gyel együtt járó nehézségre, azonban egy csomó új fejlesztést is hordoz a jövő Internete számára. A fejlődés főképpen az útválasztás, az önműködő hálózati beállítás, a biztonság és a hordozhatóság területeire terjed ki.

Az IPv6 a lehetőségek óriási bőségkosarát jelenti, amelyben a címzés csupán az egyik, bár a leglátványosabb elem. A címzésnek sok figyelmet szenteltek, ez azonban csak az egyik kiemelkedő témakör, amelyet a tervezők megragadtak. Más rendszeradottságokat a méretezhető hálózati elrendezések, a fokozott biztonsági igények, az adatépség, az összevont szolgáltatások, az önműködő beállítás, a hordozható megoldások, az adatok üzenetszórásos közvetítése, és a globális gerincvezeték szintjén hatékonyabb halmozott hálózati útválasztás iránti igények alapján fejlesztettek.

Írásunk az IPv6-tal kapcsolatos munkálatoknak a kanadai Montrealban található Ericsson Kutatóközpontban működő Open Systems Labnél folytatott Advanced Research on Internet E-Servers, röviden az ARIES kezdeményezés részleteit világítja meg. Az ARIES kezdeményezés 2000 januárjában indult útjára, amely céljául fürtözött Internet-kiszolgálótelep elkészíthetőségéhez szükséges műszaki háttér megtervezését és a prototípus elkészítését tűzte ki. A próbagép távközlési fokozatú (telecom-grade), és alaprendszerként Linuxot és nyílt forrású rendszert fog használni.

Az IPv6 létfontosságú módszer, amely pontosan ilyen számítógéptelepek és távközlési rendszerek támogatására hivatott. Az Ericsson jövőképében minden nem helyhez kötött felhasználó „állandóan a hálózatra kapcsolódik és folytonosan elérhetőként" szerepel. Ez az új IP-technológia használatához kapcsolódó újfajta szolgáltatás számos szolgáltatás beindítására teremt lehetőséget az internetszolgáltatóknak és a hálózatüzemeltetőknek.

Az internetfelhasználók számának gyors növekedésével együttjáró vezeték nélküli internetkészülékek várható gyarapodása azonban méretezhető és rugalmas IP-módszert igényel, amely képes teret adni a gyors növekedésnek. Éppen ezért alapvető jelentőségű az IPv6 kulcsszerepének elismerése, és a „látomásban" szereplő állandó hálózati kapcsolat és folytonos elérhetőség fontosságának felismerése. Az új szolgáltatások, mint amilyen az IP-multimédia is, az elérhetőség érdekében az egész földgolyóra kiterjedő egyedi címzés használatát tételezik fel. Az IPv6 a hozzá tartozó nagy címtartománnyal együtt minden készülék számára ilyen globálisan egyedi címeket fog biztosítani.

Cikkünk a nyílt forrású IPv6-tal kapcsolatos kezdeményezéseket térképezi fel. A cél a jelenleg hozzáférhető Linux IPv6-megvalósításokkal való kísérletezés és ajánlások megfogalmazása, hogy a közel távközlési szintű kiszolgálótelepünkön működő Linux-rendszerünk számára melyik változatot részesítsük előnyben.

Az ajánlásoknak több vizsgálati szempontra kell épülniük, úgymint alkalmazásfejlesztési sebesség, a szabványoknak való megfelelés és a más alkalmazásokhoz mért teljesítmény.

Nyílt forráskódú IPv6-kezdeményezések

Az első lépés a Nyílt Forráskód Közösségének áttekintése volt, továbbá jelentés készítése az IPv6-megvalósítások biztosítását célzó kezdeményezésekről – továbbfejlesztve a már létező alkalmazásokat, illetve kipróbálási és ellenőrzési lehetőséget teremtve más alkalmazások számára.

A WIDE IPv6 Working Group (a továbbiakban IPv6 WG) a WIDE-kezdeményezés része. 1995-ben Japánban az IPv6-tal való kísérletezés és telepítés céljából indították útjára. 1995 végén az IPv6 WG már több egymástól független változattal rendelkezett, és más rendszerekkel való együttműködési bemutatókat tartottak. Amint a meghatározásokat átvizsgálták és az együttműködés gyakorivá vált, úgy tűnt, hogy az IPv6 WG az IPv6-vermeket nem tudta egymástól függetlenül hatékonyan megvalósítani.

A megvalósítások erősségeinek összefogására a WIDE kezdeményezés alvállalkozásként útnak indította a KAME kezdeményezést. Bár az IPv6 WG és a KAME tevékenysége között akad átfedés, az IPv6 WG főképp műszaki és újítási kutatást folytat, a KAME ugyanakkor a megvalósításokért felelős. A WIDE IPv6 WG célkitűzései az IPv6-megvalósítások kialakításában és az ezekhez kapcsolódó programok biztosításában, az IPv6 ipari termelési környezetben való telepítésének ösztönzésében és az IPv4-ről az IPv6-ra való áttérési módszer kidolgozásában foglalható össze. A fentieken kívül a kezdeményezés céljául tűzte ki az IPv6-hálózatokhoz tartozó módszerek és szaktudás kifejlesztését is.

Az IPv6 a lehetőségeket tartalmazó hatalmas csomag, amelyben a címzés a legkézzelfoghatóbb elem. A kezdeményezés várható eredménye az RFC-knek megfelelő, ingyenes IPv6/IPSec-forráskód. A KAME kezdeményezés Japánban hét vállalatot egyesítő közös erőfeszítés, amely a BSD-változatok (FreeBSD, OpenBSD, NetBSD, BSD/OS) számára ingyenes, megbízható termék életre hívása végett jött létre, és különösen az IPv6-ot, illetve az IPSec-et veszi célba.

A kezdeményezést azért hozták létre, hogy elkerülje a szükségtelen fejlesztési átfedéseket, valamint jó minőségű és

magas színvonalú szolgáltatásokat felvonultató alkalmazásokat hozzon létre. A kezdeményezés még 1998-ban kezdte meg működését, és a tervek szerint 2002 márciusáig tevékenykedik. A vezető kutatók a következő társaságoktól gyűltek egybe: Fujitsu Limited, Hitachi Ltd., IIJ Research Laboratory, NEC Corporation, Toshiba Corporation és Yokogawa Electric Corporation. Arra vállalkoztak, hogy teljes munkaidőben dolgoznak az IPv6-vermen, tehát fő feladatukként kezelik. Tervük, hogy a BSD szerzői jog alapján a lehető legjobb hálózati kódot fejlesszék ki, és eredményeiket nyilvános programként jelentessék meg.
A vállalkozás várható eredménye az IPv6/IPSec számára a kiváló forráskód lesz – szabad programként megjelentetve –, amely eredetileg a WIDE Hydrangea IPv6/IPSec-megoldáson alapul, ez lesz a XXI. század haladó szintű internetmegoldásának kiindulópontja.
A TAHI kezdeményezés 1998 októberében Japánban kezdte meg működését a Tokiói Egyetem (University of Tokyo), az YDC Co. vállalat és a Yokogawa Electric Co. közötti vállalkozásként. Elképzelésük az, hogy az IPv6 számára ellenőrzési módszert fejlesszenek ki, és ezeket az együttműködési és megfelelési próbák kifejlesztésén és kutatásán keresztül vizsgálják meg.
A csoport a minőségfejlesztő oldalon a KAME kezdeményezéssel közösen dolgozik a TAHI kezdeményezés által fejlesztett módszer kínálásában és a fejlesztési hatékonyság javításában. Fontos megjegyezni, hogy a TAHI kezdeményezés a WIDE kezdeményezéstől is támogatást kap. A TAHI kezdeményezés az alábbi eredményeket hozta:

- Megfelelési próbák: a kezdeményezés a megfelelési próbaprogram készleteit kéthavonta bocsátja ki. Megeshet, hogy kibocsátási dátumának időzítése egybeesik a KAME megbízható változatának megjelenésével.
- Együttműködési próbák egyszerű hálózatokban: ezek az ellenőrzések azt vizsgálják, hogy vajon a célgép tud-e egyedi hálózatokban is dolgozni. Mivel e tesztet korlátozott környezetekben fogják folytatni, ezt lesz az együttműködési próbák első lépcsője.
- Együttműködés többféle IPv6-megvalósítást tartalmazó környezetben: ezek a vizsgálatok a való világgal folytatott együttműködést ellenőrzik.
- Próbaforgatókönyvek és vizsgálati eszközök.

A vállalkozás eredményei a nyilvánosság számára szabadon és ingyenesen hozzáférhetők. Az USAGI- (UniverSAl Play Ground for IPv6) kezdeményezés a Linux számára készített IPv6-protokoll ipari minőségének előállításán dolgozik, együttműködve a WIDE kezdeményezéssel, a KAME kezdeményezéssel, a TAHI kezdeményezéssel és a Linux IPv6 Felhasználói Csoportjával. Az USAGI kezdeményezés IPv6-megvalósítását a későbbiekben tárgyaljuk.
Az IPv6-DRET kezdeményezés az IPv6 nyilvános Linux-megvalósítása, amelyet a DGA/DRET (Francia Katonai Kutató Ügynökség) alapított, és az INRIA Sophia-Antipolis és az LIP6 Paris szervezetekkel együtt közösen fejlesztett.
Az IPv6-DRET-változatot a 2.1-es Linux-rendszermagra építették, és a megvalósítás célja az volt, hogy a szolgáltatás minőségéhez (QoS) megvizsgáljon bizonyos algoritmusokat. A célkitűzés nem a teljes körű alkalmazás biztosítása volt, hanem a QoS-hez kapcsolódó bizonyos algoritmusok kipróbálása.
Mivel az útválasztási kérdések az IPv6-tal kapcsolatos legfontosabb kutatási területek között szerepelnek, a hálózati veremben az IPv6-os gazdagépnek és útválasztó gépeknek egyaránt be kell kerülniük.
Az útválasztó-meghatározás megvalósítása mellett sok energiát fektettek a GateD-ben levő RIPng-fejlesztésébe is.
A kezdeményezés befejezte működését, a fejlesztés abbamaradt és a programkód elavulttá vált.
A Linux IPv6 RPM kezdeményezés rpm-csomagokat készít rpm-alapú rendszerek számára, például a Red Hat Linux, TurboLinux és a Caldera OpenLinux számára. Az IPv6-hálózatokra kapcsolódáshoz szükséges programokat és eszközöket tartalmaz, a kezdeményezés célja pedig az IPv6-hálózatokhoz történő kapcsolódást megkönnyítő csomagok fejlesztése.
A Debian IPv6-kezdeményezés a Debian kezdeményezés része, és célja egyes Debian-csomagok átalakítása úgy, hogy IPv6-megfelelőkké váljanak.
Amint talán a fentebb felsorolt Linux-kezdeményezések közül is kitűnt, a Linux számára mindössze két változat kínált komoly IPv6-megvalósítást: a Linux-rendszermagra épülő megvalósítás és az USAGI kezdeményezés.
Az alábbi szakaszban részletesen elemezzük e két kezdeményezést, illetve megvalósítást, és az IPv6 Linux-rendszerbe való telepítésüket is megvizsgáljuk, vagyis hogy melyik változat felel meg leginkább az IPv6-előírásoknak.

Az USAGI kezdeményezés a Linux számára ipari minőséget elérő IP-verem előállítását és az IPv6-megfelelő hálózat kísérleti működtetését célozza meg – a kifejlesztett programot a KAME , a WIDE és a TAHI kezdeményezésekkel szoros együttműködésben használja fel.
A csoport számos magán- és tudományos szervezetből tevődik össze, és a Linux számára készített IPv6 népszerűsítését és telepítését végzi. A csoport tagjai a WIDE kezdeményezés, a CRL, a GLUON PARTNERS Co. Ltd., az INTEC Inc., a Toshiba Corporation, a Hitachi Ltd., NTT Software Corporation, a Yokogawa Electric Corporation, a Tokiói Egyetem és a Keio Egyetem.
2002. március 31. – a létező Linux-rendszermag saját IPv6 protokollveremmel rendelkezik, de a TAHI kezdeményezés értékelési eredményei alapján az IPv6 jelenlegi megvalósításának minősége elmarad a más operációs rendszerekben szereplő változatokétól, például a FreeBSD-étől vagy akár a Microsoft Windows 2000-étől.
A kezdeményezés az IPv6-rendszer fejlesztése és telepítése tekintetében más kutató-fejlesztővállalkozásokkal és szervezetekkel is együttműködik.
Az USAGI kezdeményezés munkáját összehangolja a KAME kezdeményezéssel – ennek az együttműködésnek a gyümölcse egy IPv6-változat a BSD Unix-rendszer számára – és a TAHI kezdeményezéssel, ez utóbbi termései az IPv6-próbák, értékelési meghatározások és más eszközök.
Az USAGI kezdeményezés jelentős mértékben a KAME-programkódra támaszkodik, amely tulajdonképpen egy a különböző BSD-rendszerek számára készített IPv6-verem, és erőfeszítéseik többsége a verem helyesbítésének, illetve a Linux-rendszerbe történő minél jobb illesztésének irányába mutat. Az USAGI kezdeményezés rendelkezik a Linux-rendszermaghoz használható IPv6-megvalósítással.
A kezdeményezés eredménye a Linux-rendszermaghoz illeszkedő ingyenes IPv6-verem és a ***glibc*** könyvtárban sok fejlesztést megért alkalmazói programfelületek.
A kezdeményezés második, üzembiztos változata 2001. február 5-én jelent meg, további programkódfrissítések kéthetente szerezhetők be.

A Linux-rendszermag és az IPv6

A Linux-rendszermag saját IPv6-megvalósítással rendelkezik. Mint azonban már korábban említettük, a TAHI kezde-

	Linux2.4.5	USAGI
unrecognized_next_header()	IGEN	IGEN
payload_length_zero()	IGEN	IGEN
next_header_zero()	RÉSZBEN	IGEN
header_option_processing_order_a()	NEM	IGEN
header_option_processing_order_b()	RÉSZBEN	IGEN
header_option_processing_order_c()	IGEN	IGEN
header_option_processing_order_d()	NEM	IGEN
option_processing_single_option_a()	NEM	IGEN
option_processing_single_option_b()	IGEN	IGEN
option_processing_single_option_c()	NEM	IGEN
option_processing_single_option_d()	NEM	IGEN
option_processing_single_option_e()	NEM	IGEN
option_processing_single_option_f()	IGEN	IGEN
option_processing_many_options_a()	IGEN	IGEN
option_processing_many_options_b()	IGEN	IGEN
option_processing_many_options_c()	IGEN	IGEN
option_processing_many_options_d()	NEM	IGEN
option_processing_many_options_e()	NEM	IGEN
routing_header_a()	NEM	NEM
routing_header_b()	NEM	NEM
routing_header_e()	NEM	NEM
routing_header_f()	NEM	NEM
routing_header2_a()	NEM	NEM
routing_header2_b()	NEM	NEM
routing_header2_c()	NEM	NEM
reassembly_frag_entry_a()	NEM	NEM
reassembly_frag_entry_b()	NEM	NEM
reassembly_frag_entry_c()	NEM	NEM
reassembly_frag_entry_d()	NEM	NEM
reassembly_time_exceeded_a()	NEM	NEM
reassembly_time_exceeded_b()	NEM	NEM
reassembly_time_exceeded_c()	NEM	NEM
reassembly_time_exceeded_d()	NEM	NEM
fragment_header_m_bit()	NEM	NEM
max_reassembly_size_exceeded()	NEM	NEM
stub_frag_header()	NEM	NEM
frag_multiple_m_bit_zero_a()	NEM	NEM
frag_multiple_m_bit_zero_b()	NEM	NEM
reassembly_unfragmentable_header_a()	NEM	NEM
reassembly_unfragmentable_header_b()	NEM	NEM
reassembly_unfragmentable_header_c()	NEM	NEM
no_next_header_a() // TR1, TR2	NEM	NEM
no_next_header_b()	NEM	NEM

1. kép Alapvető meghatározás

	Linux2.4.5	USAGI
redirect_handling_a	IGEN	RÉSZBEN
redirect_handling_b	RÉSZBEN	RÉSZBEN
redirect_handling_c	IGEN	IGEN
redirect_handling_d	RÉSZBEN	RÉSZBEN
redirect_handling_e	IGEN	IGEN
redirect_validation_1_a	IGEN	IGEN
redirect_validation_1_b	IGEN	IGEN
redirect_validation_1_c	IGEN	IGEN
redirect_validation_1_d	IGEN	IGEN
redirect_validation_1_e	IGEN	IGEN
redirect_validation_1_f	NEM	NEM
redirect_validation_2_a	IGEN	IGEN
redirect_validation_2_b	IGEN	IGEN
redirect_validation_2_c	RÉSZBEN	IGEN
redirect_validation_2_d	IGEN	IGEN
unexpected_option_a	IGEN	IGEN
unexpected_option_b	IGEN	IGEN
unexpected_option_c	IGEN	IGEN
unexpected_option_d	IGEN	IGEN
no_dce_redirect_a	IGEN	IGEN
no_dce_redirect_b	IGEN	IGEN
neighbor_cache_redirect_a	IGEN	IGEN
neighbor_cache_redirect_b	IGEN	IGEN
neighbor_cache_redirect_c	IGEN	IGEN
redirected_next_hop_unreachable	IGEN	IGEN

2. kép Önműködö címbeállítás

	Linux2.4.5	USAGI
duplicate_address_detection_a()	IGEN	IGEN
duplicate_address_detection_b()	IGEN	IGEN
duplicate_address_detection_c()	IGEN	IGEN
duplicate_address_detection_d()	IGEN	IGEN
preferred_greater_than_valid()	IGEN	IGEN
ra_from_global_source()	NEM	NEM
ra_oa_bits_reset()	NEM	NEM

3. kép Átirányítás

ményezés alapján bebizonyosodott, hogy elmarad más megvalósítások mögött. Ez nem nagy meglepetés, hiszen jelentősebb fejlesztés már egy ideje nem zajlik. További tényként említendő meg, hogy ***Pedro Roque***, a Linux-rendszermag IPv6 programkódjának írója kivált a közösségből. Azóta ***Alexey Kuznetsov*** és társai az évek alatt meglehetősen kevés javítást végeztek rajta.
Az USAGI kezdeményezés csak apró hibajavításokkal jelentkezett, de a veremnek a rendszermaggal való teljes egybeolvasztása még mindig nyitott kérdés.

Megfelelési vizsgálatok

Ezen a ponton laboratóriumunkban két próbahálózatot működtetünk. Az egyik hálózatban USAGI IPv6-tal ellátott Linux-gépek működtek, míg a másikon linuxos gépek az IPv6-veremmel kiegészített rendszermaggal. Sokat foglalkoztunk a telepítés végrehajtásával, az útválasztást és tunnelinget érintő kérdésekkel.
A kérdés továbbra is az maradt, hogy melyik változatot fogadjuk el.
Ahhoz, hogy erre a kérdésre tárgyilagosan tudjunk válaszolni, a budapesti Ericsson Kutatóközpont megfelelési próbákat végzett a legfrissebb hivatalos rendszermaggal – ez a próbák idején a 2.4.5-ös volt, illetve a próbában az USAGI IPv6-megvalósítás is szerepelt (a 2.4.0-s változat alapján).
A vizsgálatokat a University of New Hampshire InterOperability Lab IPv6 Ellenőrzési leírására alapoztuk (lásd a ***Kapcsolódó címeket***).
A vizsgálatok értékelése során az alábbi minősítéseket használtuk:

IGEN – (megfelelt) a megvalósítás az adott részfeladatot megoldotta.
NEM – (hibás) a megvalósítás képtelen volt a kitűzött részfeladatot megoldani.
RÉSZBEN – döntetlen ítélet született, azaz nem lehetett eldönteni, hogy a megvalósítás képes-e a próbának megfelelni. Ha például a próbafolyamat három kérés-válaszlépésből áll, és a tesztelő második kérésére sem érkezik meg a válasz, a minősítés döntetlen.
A megfelelést vizsgáló laboratórium négy ellenőrzésfajtát folytatott le: alapvető meghatározás, önműködő címbeállítás, átirányítás és környezetfelfedezés. Az alábbiakban részletesen kifejtjük e próbákat, és ismertetjük a kapott eredményeket.
Alapvető meghatározás: ez a vizsgálatsorozat az IPv6 alapvető meghatározásaira vonatkozik. Az alapmeghatározás megadja az IPv6 fejlécet és a kezdeti értékadás révén meghatározott IPv6-fejléceket és a választható lehetőségeket. Elemzi a csomagmérettel kapcsolatos gondokat, a forgalomosztályok és folyamatjelzők jelentéstanát, és az IPv6-nak a felsőbb protokollrétegekre gyakorolt hatását (lásd az *1. képet*).
Önműködő címbeállítás: ezek a próbák a cím önálló beállítását vizsgálják az IPv6-nál. Jellegük olyan, hogy az IPv6 állapotnélküli önbeállítás-meghatározásnak való megfelelését vizsgálja (*2. kép*).
Átirányítás: az átirányítási próbák az IPv6-környezet felfedezési meghatározás átirányítási lehetőségére vonatkozik. Az átirányított üzeneteket az útválasztó küldi, hogy az adott úti cél eléréséhez egy gazdagépet egy jobb első ugrásbeli (first-hop) útválasztóhoz irányítson, vagy a gazdagépnek mondja meg, hogy a végcél maga is egy szomszéd, tehát már mutat rá hivatkozás (*3. kép*).
Környezetfelfedezés: e vizsgálatok az IPv6-nál a szomszéd felfedezésére vonatkoztak. A környezetfelfedező protokollt az elemek használják – a gazdagépek és az útválasztók –, hogy az ismert kapcsolódó hivatkozásokon levő szomszédok számára meghatározzák a kapcsolati réteget, és mielőbb kitöröljék a gyorstárban lévő immár érvénytelen adatokat. A gazdagépek szintén a környezeti felfedezést használják a közelben levő útválasztók felkutatására, amelyek a kapott csomagokat majd ezek nevében továbbítják. Végül a csomópontok is használják a protokollt, hogy nyomon kövessék, mely szomszédok érhetők el, és melyek nem. Ha egy útválasztó vagy egy útválasztóhoz vezető útvonal elérése sikertelen, a gazdagép más használható útvonalak után néz (*4. és 5. kép*).
Az IPv6 az a kulcsfontosságú módszer, amely képes az állandóan a hálózatra kapcsolódó, nagyszámú felhasználóról képet alkotni.
Ilyen eredményekre alapozva megállapíthatjuk, hogy az USAGI megvalósítás a Linux-rendszermagba épített változatnál kedvezőbb eredményeket hozott: több próbán szerepelt sikeresen, kevesebb tesztnél született hibás eredmény, és az eldönthetetlen esetek száma is kevesebb volt, mint a rendszermagba épített változatnál.

Most már tudjuk a leckét

Jellemző módon az ilyen feladatok végrehajtása közben sokat lehet tanulni, és a szakértelem is gyarapodik. Mindig akadnak azonban olyan tudnivalók, amelyekre csak a munka végzése közben lehet szert tenni, és nagyon is megéri felhívni rájuk a figyelmet. Mindig kifizetődő, ha a problémákat egyszerűsítjük. Például nem lehetett tudni, hogy a `ping6` miért volt képes a 2.4.5-ös rendszermag összeomlását előidézni, miközben három webkiszol-

	Linux2.4.5	USAGI
ND_on_link_determination_a	IGEN	IGEN
ND_on_link_determination_bc	IGEN	IGEN
ND_on_link_determination_d	IGEN	IGEN
ND_resolution_wait_queue	NEM	NEM
ND_add_default_router	IGEN	IGEN
ND_default_router_switch	NEM	NEM
ND_host_ra_processing_lifetime_a	NEM	NEM
ND_host_ra_processing_lifetime_b	IGEN	IGEN
ND_reachable_time	IGEN	IGEN
ND_neighbor_cache_ra_a	NEM	RÉSZBEN
ND_neighbor_cache_ra_b	IGEN	NEM
ND_neighbor_cache_ra_c	NEM	NEM
ND_neighbor_cache_ra_d	NEM	IGEN
ND_prefix_on_link_bit	RÉSZBEN	IGEN
ND_host_prefix_list_a	IGEN	IGEN
ND_host_prefix_list_b	IGEN	IGEN
ND_host_prefix_list_c	IGEN	IGEN
ND_host_prefix_list_d	NEM	NEM
ND_prefix_invalidation	NEM	NEM
ND_default_router_selection	RÉSZBEN	RÉSZBEN
ND_neighbor_solicitation_origination_a	IGEN	IGEN
ND_neighbor_solicitation_origination_b	RÉSZBEN	RÉSZBEN
ND_neighbor_solicitation_origination_c	IGEN	IGEN
ND_neighbor_solicitation_handling_a	IGEN	IGEN
ND_neighbor_solicitation_handling_b	IGEN	IGEN
ND_neighbor_solicitation_handling_c	IGEN	IGEN
ND_neighbor_solicitation_handling_d	IGEN	IGEN
ND_no_neighbor_cache_na	NEM	NEM
ND_neighbor_cache_incomplete_b	NEM	NEM
ND_neighbor_cache_incomplete_c	NEM	NEM
ND_neighbor_cache_incomplete_d	NEM	NEM
ND_neighbor_cache_incomplete_e	NEM	NEM
ND_neighbor_cache_stale_a	IGEN	IGEN
ND_neighbor_cache_stale_b	NEM	NEM
ND_neighbor_cache_stale_c	NEM	NEM
ND_neighbor_cache_stale_d	NEM	NEM
ND_neighbor_cache_stale_e	NEM	NEM
ND_neighbor_cache_stale_f	NEM	NEM
ND_neighbor_cache_stale_g	IGEN	IGEN
ND_neighbor_cache_stale_h	IGEN	IGEN
ND_neighbor_cache_delay_a	IGEN	IGEN
ND_neighbor_cache_delay_b	NEM	NEM
ND_neighbor_cache_delay_c	NEM	NEM
ND_neighbor_cache_delay_d	NEM	NEM
ND_neighbor_cache_delay_e	NEM	NEM
ND_neighbor_cache_delay_f	NEM	NEM
ND_neighbor_cache_delay_g	IGEN	IGEN
ND_neighbor_cache_delay_h	IGEN	IGEN
ND_neighbor_cache_probe_a	IGEN	IGEN
ND_neighbor_cache_probe_b	NEM	NEM
ND_neighbor_cache_probe_c	NEM	RÉSZBEN
ND_neighbor_cache_probe_d	IGEN	NEM
ND_neighbor_cache_probe_e	RÉSZBEN	RÉSZBEN
ND_neighbor_cache_probe_f	RÉSZBEN	RÉSZBEN
ND_neighbor_cache_probe_g	IGEN	IGEN
ND_neighbor_cache_probe_h	NEM	NEM
ND_neighbor_cache_reachable_a	IGEN	IGEN
ND_neighbor_cache_reachable_b	NEM	NEM
ND_neighbor_cache_reachable_c	NEM	NEM
ND_neighbor_cache_reachable_d	NEM	NEM
ND_neighbor_cache_reachable_e	NEM	NEM
ND_neighbor_cache_reachable_f	NEM	NEM
ND_neighbor_cache_reachable_g	IGEN	IGEN
ND_neighbor_cache_reachable_h	IGEN	IGEN
ND_na_r_bit_change_a	RÉSZBEN	RÉSZBEN
ND_na_r_bit_change_b	RÉSZBEN	RÉSZBEN
ND_na_r_bit_change_c	RÉSZBEN	RÉSZBEN

4. kép Környezetfelfedezés

	Igen		Nem		Nem értelmezett	
	USAGI	Linux	USAGI	Linux	USAGI	Linux
Alapkövetelmények	18	8	25	33	0	2
Önműködő címbeállítás	5	5	2	2	0	0
Átirányítás	21	21	1	1	3	3
Szomszédság feltételezése	28	28	30	31	9	8
Összes	72	62	58	67	12	13

5. kép A próbák eredményei

gáló-alkalmazást futtattunk: az IPv6 fogadására alkalmas javítófolttal kiegészített Apache, Jigsaw és Tomcat kiszolgálókat, valamint az alfaváltozatú Java Merlin 1.4-et. Minden kitisztult azonban akkor, amikor ezeket a kiszolgálókat és Java Virtual Machine-t kikapcsoltuk. Ezenkívül teljes megvalósítás beépítése sokkal könnyebb és rugalmasabb is, mint a javítófoltok telepítése. Némely esetben programhibával találtuk szemben magunkat, és nem volt könnyű megállapítani, honnan származtak. Vajon az alkalmazás IPv6-foltjai, az USAGI IPv6 rendszermagfoltok, vagy az IPv6 foltozott bináris állományai vezettek a hiba jelentkezéséhez? A probléma leegyszerűsítése sokat segített, de azt is bebizonyította: lehet, hogy egyszerűbb egy átfogó megvalósítást beépíteni, mint a letöltött foltokat egyenként beilleszteni. Feltétlenül meg kell említenünk, hogy a Nyílt Forráskód Közössége nagyon kellemes benyomást tett ránk. A nyílt forráskóddal dolgozó emberek minden tőlük telhetőt megtesznek, hogy a felmerült programhibák mielőbb elháríthatók legyenek, és elektronikus leveleinkre is gyorsan válaszolnak. Minthogy a linuxos gépeinken használt programok többsége nyílt forrású, kérdéseinkkel visszatérünk a közösséghez, és gond nélkül közvetlenül a program fejlesztőjétől kapunk támogatást – ezt a fontos tényezőt el kell ismernünk.

Összegzés

A Linux-rendszermagba beépített IPv6-megvalósításon a 2.4.0-s változat megjelenése óta semmilyen jelentősebb fejlesztés nem történt. Sok esetben IPv6-forgalommal még a rendszermag összeomlását is képesek voltunk előidézni. Másrészről viszont az USAGI a Linux számára olyan IPv6-megvalósítást biztosít, amelyet a FreeBSD IPv6-veremből ültetett át, és amely jelenleg a legjobb IPv6-megvalósítás. Megbízhatóbbnak bizonyult és kedvezőbb megfelelési próbaeredményeket hozott, mint a linuxos rendszermag-megvalósítások. Az a tanulság szűrhető le, hogy az USAGI-változat érettebb a rendszermagba épített megvalósításnál, több szolgáltatást biztosít és az előírásoknak való nagyobb mértékű megfelelés jellemző rá. Ha több tényező alapján szeretné eldönteni, melyik IPv6-megvalósítást telepítse linuxos kiszolgálógépeire, mérlegelnie kell például a fejlettségi állapotot, a kiszolgálón dolgozó emberek számát, az ipar és tudományos világ felől érkező támogatást, a legfrissebb RFC-knek való megfelelést – mindezek alapján valószínűleg az USAGI kezdeményezés IPv6-megvalósítását fogja választani. Mindamellett mindkét alkalmazás kapcsán egy fontos aggodalom merült fel. Sajnos nincs teljes leírás, amely a támogatott RFC-ket és az egyes tervezési döntések mögötti motivációkat részletezné. Reménykedem benne, hogy a Linux-közösség és az USAGI kezdeményezés szorosabbra fűzi az együttműködés szálait, és a Linux számára nagyon megbízható és hatékony IPv6-megvalósítással áll elő. Hiszek abban, hogy az eredmény egy jól megírt, rendszermagba ágyazott IPv6-megvalósítás lesz – egy nagyon óhajtott sikerrecept a jövő IPv6-kiszolgálói számára, amely a nem helyhezkötött Internet terjedéséhez is hozzá fog járulni.

Köszönetnyilvánítások

Köszönet illeti a kanadai Ericsson Kutatóközpontban működő Open Systems Labet a cikk megjelenéséhez adott hozzájárulásukért, továbbá ***Marc Chatel***-et, ***Bruno Hivert***-et és ***David Gordon***-et a kanadai Ericsson Kutatóközpontban segítségükért és laboratóriumban nyújtott támogatásukért, és a magyarországi Ericsson Kutatóközpontot a megfelelési próbák laboratóriumában elvégzett tesztekért.

Linux Journal április, 96. szám

Ibrahim Haddad jelenleg a kanadai Montrealban működő Ericsson Corporate Research Unitnál dolgozik kutatóként. Főként az átviteli osztályba (carrier-class) tartozó kiszolgálógépek kutatásával foglalkozik – valós idejű, tisztán IP-hálózatokon. Elérhető a Ibrahim.Haddad@Ericsson.com címen

Kapcsolódó címek

KAME Projekt ➲ www.kame.net
Linux-rendszermag ➲ www.kernel.org
TAHI Project ➲ www.tahi.org
USAGI Project ➲ www.linux-ipv6.org
WIDE v6 Working Group ➲ www.v6.wide.ad.jp

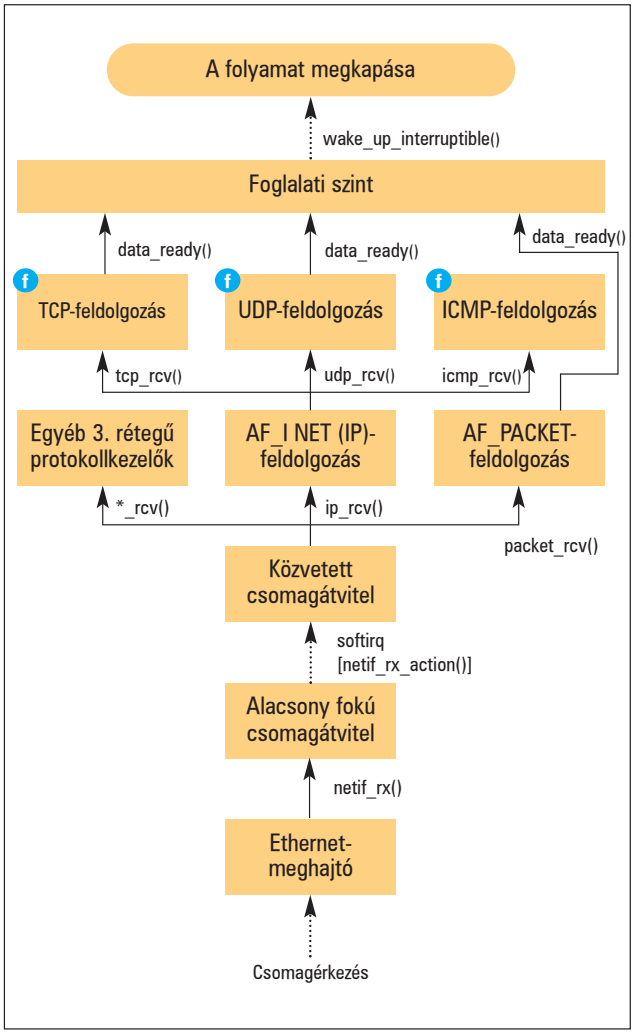
A Linux csomagszűrő felépítése (2. rész)

Gianluca elmeséli, hogy a rendszermagon áthaladva a TCP-feldolgozó végül hogyan fogja el a csomagokat.

Sorozatunk előző részben azt tárgyaltuk, hogyan jut el egy csomag a fizikai szintről – a hálózati kábelről – a felsőbb szinteken található hálózati verembe. A csomagot egészen a 3. szintig kísértük el, ahol az IP-nek már nincs több dolga, és a feldolgozást a TCP vagy az UDP veszi át. Ebben a cikkben utazásunkat a 4. szint elemzésével, a `PF_PACKET`-protokoll felépítésével és a foglalati szűrő kapcsolataival fejezzük be.
Eltekintve az IGMP- és ICMP-feldolgozástól, amelyeket a rendszermag dolgoz fel, a csomag útja az alkalmazás felé kétféleképpen végződhet: vagy a `tcp_v4_rcv()`, vagy a `udp_rcv()` függvénynek adódik át. A TCP feldolgozása egy kicsit nyakatekert, mivel a protokoll maga egy FSM (Final State Machine, vagyis véges állapotú gép), és egy egész sor közbenső állapota lehetséges. Gondoljunk csak a TCP-foglalat különböző állapotaira: `listening` (kapcsolatra vár), `established` (a kapcsolat létrejött), `closed` (a kapcsolat véget ért), `waiting` (a kapcsolat lezárása folyamatban) stb.
Hogy leegyszerűsítsük a leírást, a folyamatot a következő lépésekre bontjuk fel:

- A `tcp_v4_rcv()` függvény (***net/ipv4/tcp_ipv4.c***) ellenőrzi a TCP-fejléc sértetlenségét.
- A `__tcp_v4_lookup()` függvény felkutatja azt a foglalatot, amely erre a csomagra vár.
- Ha nem találja meg, megteszi az ilyenkor szükséges lépéseket (többek között utasítja az IP-t, hogy egy ICMP-hibaüzenetet hozzon létre).
- Ha megvan a foglalat, a csomagot továbbadja a `tcp_v4_do_rcv()`-nek, mely kézbesíti azt (`sk_buff`) és a hozzá tartozó foglalati kiépítést.

Mindemellett ez a függvény felelős minden egyéb teendő elvégzéséért, mely a csomag állapotától függően szükséges. A csomagszűrő szempontjából a TCP-feldolgozás során meghívott legutolsó függvény az érdekes, amelyről szó is esett. Az `sk_filter()` függvény a `tcp_v4_do_rcv()` hívás legelején kerül meghívásra, ami – mint majd látni fogjuk – minden csomagszűrővel kapcsolatos varázslatért felelős. Honnan tudja a rendszermag, hogy egy adott foglalatra érkező csomag esetén meg kell hívnia a csomagszűrőt? Ez az adat a foglalathoz tartozó foglalati felépítésben van tárolva. Ha a `setsockopt()` rendszerhívással valamilyen szűrőt kapcsolunk, a felépítésben a megfelelő érték valamilyen igaz értéket vesz fel, és a TCP-kezelőhívás tudni fogja, hogy meg kell hívnia az `sk_filter()`-t.
Azok számára, akik már járatosak a foglalatok programozásában és fel tudják idézni, hogy kapcsolat érkezésekor a kiszolgálóalkalmazásból egy új példány `fork()`-olódik, megemlítjük, hogy a szűrőt először egy kapcsolatra váró (`listening`) foglalatra csatlakoztatjuk. Ezután új kapcsolat érkezésekor a szűrő lemásolódik az újonnan létrejött foglalatba.
Amennyiben érdekelnek a részletek, vess egy pillantást a `tcp_create_openreq_child()` függvényre a ***net/ipv4/tcp_minisocks.c*** fájlban.
A csomagkezelőhöz visszatérve: a csomag további sorsa a szűrő kimenetétől függ; ha a csomag megfelel a szűrőben található feltételnek, akkor a feldolgozás tovább folyik, máskülönben a



csomag eldobódik. Ezen túlmenően a csomagszűrő meghatározhatja azt a legnagyobb csomagméretet, amelyet feldolgozásra továbbít (a csomagok kilógó részét az `skb_trim()` függvény vágja le). A foglalat állapotától függően a csomag útja több irányban folytatódhat. Ha a kapcsolat már létrejött, a csomagot a `tcp_rcv_established()` függvény kapja meg. Ennek a függvénynek nagyon fontos feladata van: elvégzi a bonyolult TCP-visszaigazolási rendszer kezelését és a fejlécek feldolgozását, amelyekre most nem térünk ki. Egyedül a foglalathoz (`sk`) tartozó `data_ready()` függvényt említjük meg,

A memória címeinek szűrése

Család	*Foglalattípus*	*Protokollnév*	*A memória ezzel kezdi*
PF_PACKET	STOCK_RAW	Nyers csomag	Összekötőréteg-fejléc
PF_PACKET	STOCK_DGRAM	Csomag	IP-fejléc
PF_PACKET	STOCK_RAW	Nyers IP	IP-fejléc
PF_PACKET	STOCK_DGRAM	UDP	UDP-fejléc
PF_PACKET	STOCK_DGRAM	TCP	TCP-fejléc

mely meghívja a `sock_def_readable()`-t, ez a `wake_up_interruptible()` függvény segítségével felébreszti a fogadó alkalmazást.
Szerencsére az UDP-feldolgozás ennél jóval egyszerűbb. Az `udp_rcv()` függvény, miután néhány ellenőrzést elvégzett a csomagon, az `udp_v4_lookup()` függvényt hívja meg, ami megkeresi a fogadó foglalatot, és meghívja az `udp_queue_rcv_skb()`-t. Ha nem talál megfelelő foglalatot, a csomagot eldobja.
Az utóbbi függvény meghívja a `sock_queue_rcv_skb()`-t (***sock.h***), amely a csomagot a foglalat várakozó sorába helyezi. Ha ebben az átmeneti tárban már nincs több hely, a csomag eldobódik. A szűrést is ez a függvény végzi a TCP-feldolgozásnál látottakhoz hasonlóan, vagyis az `sk_filter()` segítségével. Végül pedig a `data_ready()` függvény hívódik meg, és ezzel az UDP-csomag feldolgozása véget is ért.

Mi történik a PF_PACKET-csomagokkal?

A `PF_PACKET` család külön említést érdemel. Ebben az esetben a beérkező csomagok mindenféle feldolgozás nélkül az alkalmazás foglalatába kerülnek. Ismerve a csomagfeldolgozó rendszert, amelyet az eddigiekben tárgyaltunk, megállapíthatjuk, hogy nem túl bonyolult dologról van szó.
Ha létrehozunk egy `PF_PACKET` foglalatot (lásd a `packet_create()` függvényt a ***net/packet/af_packet.c*** fájlban), a `NET_RX softirq` által használt listához egy új protokoll adódik hozzá. Az ehhez a protokollcsaládhoz tartozó csomagtípus bekerül az általános (`ptype_all`) vagy a protokolljellemző listába (`ptype_base`), fogadófüggvénye pedig a `packet_rcv()` lesz. Bizonyos okok miatt, amelyeket csak később tisztázunk, az újonnan létrehozott foglalatok címe a csomag típusának megfelelő adatszerkezetben tárolódik. Ez a cím logikailag valójában a rendszermagnak nem ehhez a részéhez tartozna, csak később, egy 4. szintbeli kódban lesz rá szükség, ahol a foglalati adatok feldolgozásra kerülnek. Ezért ebben az esetben ez az adat a bejegyzett protokoll adatmezői között hozzáférhetetlen adatként tárolódik – egy lefoglalt mező a szerkezetben, mely a protokoll működéséhez szükséges.
Ettől a pillanattól kezdve az összes gépre belépő csomag, mely túljut a szokásos fogadási eljáráson, a `net_rx_action()` futása során átadódik a `PF_PACKET` fogadófüggvényének. Ennek a függvénynek első dolga, hogy megpróbálja visszaállítani a kapcsolati szintű fejlécet, ha a foglalat típusa `SOCK_RAW` (emlékezzünk vissza a „Csomagszűrő: bájtok leszippantása a hálózatról” című írásomra a Linuxvilág 2001. június–júliusi számában, miszerint a `SOCK_DGRAM`-foglalatok nem látják a kapcsolati réteg fejlécét).
Ezt a fejlécet vagy a hálózati kártya távolítja el (vagy bármilyen hálózati eszköz, amely a csomagot elkapja), vagy a hálózati kártya eszközmeghajtója (megszakításkezelője). Hálózati kártyák esetén szinte kivétel nélkül ez utóbbi a helyzet. A kapcsolati szintű fejléc egyáltalán nem állítható vissza amennyiben a hálózati eszköz távolította el, mivel ebben az esetben az adat soha nem kerül a rendszermemóriába, és a hálózati eszközön kívül teljesen láthatatlan. A fejlécvisszaállító művelet egyáltalán nem költséges, köszönhetően a rendszermag által nyilvántartott `skbuff` átmeneti táraknak. A következő lépéssel ellenőrizhető, hogy a fogadófoglalatra van-e szűrő kapcsolva. Ez a rész egy kicsit macerás, lévén a szűrővel kapcsolatos adatok a foglalati felépítésben tárolódnak, amelyet még nem ismerünk, mivel a protokollverem legalján vagyunk. `PF_PACKET`-foglalatok esetén azonban – melyeknek kötelezően meg kell kerülniük ezt a vermet – tudniuk kell a foglalati felépítés címét. Ez megmagyarázza, hogy a foglalat létrehozása idején miért kellett a foglalati felépítés címét a protokollblokk saját részébe beírnunk; ez viszonylag tiszta módot biztosít arra, hogy a csomag fogadásakor hozzáférjünk az adathoz.
Azáltal, hogy a foglalati felépítés kéznél van, a rendszermag képes eldönteni, hogy létezik-e szűrő a foglalathoz, és hogy meg kell-e azt hívnia (az `sk_run_filter()` híváson keresztül). Mint általában, a csomag sorsáról a szűrő dönt, vagyis hogy szükség van-e rá, esetleg el kell-e dobni (`kfree_skb()`), vagy megfelelő méretűre kell-e vágni (`pskb_trim()`).
Ha a csomagot elfogadjuk, a következő lépésben az azt tartalmazó `sk_buff`-ról másolatot kell készítenünk. Erre a másolatra feltétlenül szükség van, mivel a `PF_PACKET` is felhasznál egy `sk_buff`-ot, és az esetlegesen később következő szabályszerű protokollok is. Képzeljük el azt az esetet, amikor valamely program megnyit egy `PF_PACKET`-foglalatot, miközben ugyanabban az időben egy másik program a Weben böngészik. Minden egyes csomag esetén, mely a webkapcsolathoz tartozik, a `net_rx_action()` még azelőtt meghívja a `PF_PACKET` feldolgozó eljárásait, mielőtt azok az alapértelmezett IP-protokollok feldolgozó eljárásaihoz kerülnének. Ebben az esetben a csomagból két másolat szükséges: egy a szabályszerű fogadófoglalat számára, mely a böngészőnek továbbítódik, egy pedig a `PF_PACKET`-nek, amelyet az elkapó (sniff) program kap meg. Fontos, hogy a csomagról csak azután készül másolat, miután átjutott a szűrőn. Ily módon csak azok a csomagok igényelnek további CPU-időt, amelyek a szűrő feltételeinek ténylegesen megfelelnek. Ha a csomagszűrés alkalmazási szinten valósulna meg (vagyis ha a `PF_PACKET`-et foglalati szűrő nélkül használnánk), akkor a rendszermagnak az összes beérkező csomagról másolatot kellene készítenie, ami a teljesítmény szempontjából nézve meglehetősen rossz lenne. Szerencsére a csomagmásolás mindössze annyit jelent, hogy az `sk_buff` mezőit kell lemásolnunk, és nem magát a csomagban lévő adatokat (amelyre mind az `sk_buff`-ban, mind a másolatban ugyanaz a mutató hivatkozik).
A `PF_PACKET` feladata a fogadófoglalaton a `data_ready()` függvény meghívásával ér véget, csakúgy, mint a TCP és UDP esetén is. Ezen a ponton a `recv()` vagy `recvfrom()` függvényekre várakozó program feléled, és a csomag kézbesítése a végéhez ér.

Alvó feladatok

Elgondolkoztál már azon, hogyan kerül egy feladat alvó állapotba, ha egy adott foglalaton meghívja a `recv()`, a `recvfrom()` vagy a `recvmesg()` rendszerhívások egyikét?

A folyamat valójában nagyon egyszerű: a rendszermagon belül minden `recv` függvényt a többé-kevésbé közvetlenül meghívott `sock_recv()` függvény valósít meg (***net/socket.c***). Ez a függvény pedig meghívja a `recvmsg()` függvényt, mely a foglalati felépítés protokolljellemző műveletei között van bejegyezve. Például a `PF_PACKET` protokoll esetében ez a függvény a `packet_recvmsg()`. Ez a protokollfüggő `recvmsg()` függvény többek között előbb vagy utóbb meghívja az `skb_recv_datagram()` függvényt, mely minden protokoll esetén az általános adatgramfogadás kezeléséért felelős függvény. Ez az utóbbi függvény a `wait_for_packet()` (***net/core/datagram.c***) meghívásával blokkolja az adott programot, mely ezután `TASK_INTERRUPTIBLE` állapotba, vagyis alvó állapotba lép, és egyúttal azt a foglalat várakozási sorába helyezi. A program addig pihen ott, míg egy új csomag érkezése esetén a `wake_up_interruptible()` függvény meg nem hívódik, mint azt az előző bekezdésekben is láthattuk.

Mire jó maga a szűrő?

A szűrő megvalósításának fő része a ***core/filter.c*** fájlban található, míg a `SO_ATTACH/DETACH_FILTER` `ioctl`-ek a ***net/core/sock.c*** fájlban kerülnek feldolgozásra. Kezdetben a szűrő az `sk_attach_filter()` függvény meghívásával kapcsolódik a foglalatra, amely – miután az épségét ellenőrizte (`sk_chk_filter()`) – a felhasználói szintről a rendszermag állapotterébe másolja át. Ez az ellenőrzés gondoskodik arról, hogy a szűrő parancsfeldolgozójába nehogy oda nem illő kód kerüljön. Végül a szűrő báziscíme átmásolódik a foglalati felépítés szűrőmezőjébe, ahonnan majd később meghívódik.
A csomagszűrő támogatás az `sk_run_filter()` függvényben valósul meg, mely egy `skb`-t (pillanatnyi csomagot) és egy szűrőprogramot kap. Ez utóbbi csak egy egyszerű BPF-utasításokból álló tömb, mely numerikus vezérlők és operandusok sorozatát takarja. Az `sk_run_filter()` csak egy egyszerű BPF-parancsértelmezőt valósít meg (egy virtuális CPU-t, ha úgy tetszik), teszi mindezt felettébb logikus módon; egy hosszú `switch/case` kifejezés határoz a vezérlők alapján, és végzi el a szükséges műveleteket az emulált regisztereken és a memóriában. Az emulált memóriaterület, ahol a kód lefut, természetesen a csomag adott szintnek megfelelő környezetében található (`sk->data`). A szűrő kódja egészen addig végrehajtódik, míg el nem jut egy `BPF RET` utasításig, ezután a függvény kilép.
Vedd észre, hogy az `sk_run_filter()` közvetlenül a `PF_PACKET` feldolgozó eljárásaiból hívódik meg! A foglalati szintű fogadóeljárások (TCP, UDP és nyers IP) az `sk_filter()` (***sock.h***) behúzófüggvényen haladnak keresztül, mely amellett, hogy belsőleg meghívja az `sk_run_filter()`-t, a csomagokat is a megfelelő méretűre vágja.

Kapcsolatok a csomagszűrőhöz

Utunk a rendszermag csomagszűrő függvényei között a végéhez érkezett. Érdekes levonni néhány következtetést a csomagszűrő belépési pontjaival kapcsolatban. Mint láttuk, a rendszermagon belül három jól elkülöníthető pont van, ahol a szűrő meghívódhat: a TCP és UDP fogadófüggvények (4. szint), és a `PF_PACKET` fogadófüggvénye (kettő és feledik szint). A nyers IP-csomagok kiszűrődnek, mivel ugyanazt az utat teszik meg, mint az UDP-csomagok (nevezetesen: `sock_queue_rcv_skb()`, mely adatgramok fogadására szakosodott).
Fontos megjegyezni, hogy a szűrő minden szinten az adott szintnek megfelelő környezetben fut le. Azaz ahogy egyre feljebb halad a veremben, egyre kevesebb adat áll rendelkezésére. A `PF_PACKET`-foglalatok esetén a szűrő a 2. szintnek megfelelő adatokkal rendelkezik, mely a `SOCK_RAW`-foglalatoknak a teljes kapcsolati szintű adatkereteket magában foglalja, vagy pedig 4. szintnek megfelelően a teljes IP-csomagot TCP- és UDP-foglalatoknak (alapvetően a kapuk számát, és még néhány egyéb hasznos adatot tartalmaz). Ebből következően a 4. szintű csomagszűrő értelmetlen. Természetesen az alkalmazás adatterülete mindig a szűrő rendelkezésére áll, annak ellenére is, hogy csak nagyon ritkán van rá szükség, vagy éppen semmikor. A 4. szint használhatatlanságára látható tökéletes példa az 1. és 2. listákon (➲ http://www.linuxvilag.hu/Csomagszuro webhelyen), melyeken egy egyszerű UDP-kiszolgálót figyelhetünk meg. A szűrő csak azokat a csomagokat fogadja, amelyek az ***lj*** karaktersorozattal kezdődnek (vagyis `0x6c6a` hexadecimálisan). Ahhoz, hogy ki tudjuk próbálni, fordítsuk le a programot, és nevezzük ***udprcv***-nek, majd futtassuk le. Utána fordítsuk le a 2. programot, és nevezzük ***udpsnd***-nek, végül futtassuk le azt is:

```
# ./udpsnd 127.0.0.1 "hello world"
```

Ennek hatására az ***udprcv*** semmit sem jelenít meg. Most próbáljuk meg a következő ***lj***-vel kezdődő karaktersorozattal:

```
# ./udpsnd 127.0.0.1 "lj rules"
```

Ezúttal a karaktersorozat annak rendje és módja szerint megjelent, mivel a csomag tartalma megfelel a szűrőnek. Egy másik fontos dolog, mellyel a szűrőkódolónak tisztában kell lennie, az, hogy minden egyes foglalattípus esetén (`PF_PACKET`, nyers IP, vagy TCP/UDP) más-más szűrőt kell használnia.
A szűrő a memóriahivatkozásokkor valójában relatív címzési módot alkalmaz, mely minden esetben a csomagban található adat első bájtjához igazodik. A leggyakrabban alkalmazott protokollcsaládokhoz a szűrő memóriacímeit *táblázatunk* tartalmazza.
A Linux Journal 2001. júniusi számában található cikkben a `tcpdump -dd` paranccsal nyert szűrőkód csak `PF_PACKET`-csomagokkal érvényes, mivel a 2. szintnek megfelelő szűrőkódot állít elő (ugyanis feltételezi, hogy a `0`-s cím a kapcsolati szintű keret kezdete).

Összegzés

Érdekes követni egy csomag kalandos útját a rendszermagon keresztül. Utunk során találkoztuk alapvető rendszermag-adatszerkezetekkel (`skbuff`-ok), jellegzetes programozási módszereket fedeztünk fel (felépítések használata függvényekre mutató mutatókkal a C++ objektumok hatékony választási lehetőségeként), és megismertünk néhány új 2.4-es rendszermagbeli rendszert (`softirq`-k).
Ha többet szeretnél tudni erről a témáról, fegyverkezz fel a rendszermag forrásával, keress egy jó szövegszerkesztőt, hörpints fel egy pohár kávét, majd kezdj kutakodni a kódban. A részvétel ingyenes, a szórakozás garantált!

Linux Journal március, 95. szám

Gianluca Insolvibile
a 0.99pl4-es rendszermag óta Linux-rajongó. Jelenleg hálózati és digitális videokutatással és -fejlesztéssel foglalkozik. Elérhető a g.insolvibile@cpr.it címen.

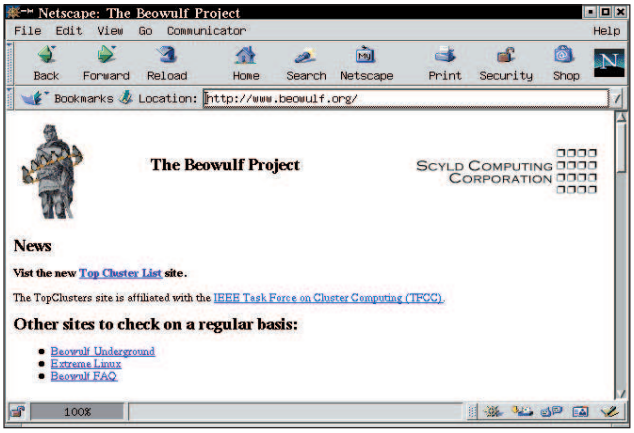
A Beowulf-tudatállapot

Mindenki találkozik OSCAR-ral, Scyld pedig rendszergazda lesz, és csodálatos géptelepeket épít.

Szeretnél linuxos géptelepet építeni? Mostanában sokan így vannak ezzel. Nem mindenki érti azonban ugyanazt a linuxos géptelep fogalmán. Ha megkérdeznénk, sokan azt válaszolnák, hogy a linuxos géptelep egyenlő a magas rendelkezésre állóságot felmutató, hibatűrő, redundáns és terheléselosztást megvalósító rendszerekkel, amelyeket az e-kereskedelmi webhelyeknél vagy az alkalmazáskiszolgálókban használnak. Mások a linuxos géptelepen a párhuzamos számításokat végző hatalmas teljesítményű Beowulf-telepeket értik. Mindannyiuknak igaza van. Ha úgy gondolod, hogy ez zavaros, próbálj csak egy kicsit elgondolkodni a bioinformatika fogalmán. Pillanatnyilag a linuxos géptelepek és a bioinformatika a leggyorsabban fejlődő technikák közé tartoznak, nem csoda, hogy fogalmaik ilyen bizonytalanok. Tisztában vagyok ezzel, hiszen hivatásszerűen foglalkozom bioinformatikával és linuxos géptelepekkel, bár e két téma kalandnak sem utolsó. Ennyit rólam, most lássuk a Beowulfot! A Beowulf ötlete ***Donald Becker*** és ***Thomes Sterling*** agyából pattant ki 1994-ben, amikor külsős munkatársként NASA-nál dolgoztak. Az ötlet hamar a nyílt forráson alapuló elosztott számítások szabványává nőtte ki magát. A recept mindössze annyi, hogy vegyünk néhány mindennapi számítógépet, kössük össze őket, telepítsünk rájuk nyílt forrású programokat, és kész is a virtuális szuperszámítógép. „Mihez kezdhetnék egy saját szuperszámítógéppel?” – kérdezheted joggal. Gyakorlatilag bármit: például MP3-zenéket gyárthatsz, vagy megfejtheted az emberi DNS-t. Ez a két alkalmazási terület kissé eltérően veszi hasznát a több processzornak. Ha a Beowulfon MP3-at gyártasz, az MP3-gyártást lényegében az egész telepen szétosztod, például minden processzorra jut egy MP3. Így nagyszámú soros feladatot párhuzamosítasz oly módon, hogy egyszerre indítod el őket, mindegyiket külön processzoron. Minden egyes MP3-feladat független a többitől, futása közben nem kell adatot cserélnie a többi MP3-folyamattal. Elvileg ezer (azonos hosszúságú) MP3 legyártása ezer (azonos sebességű) processzoron annyi ideig tart, mint amennyi ideig egyetlen MP3 elkészítése egy processzoron. Ez történne egy „tökéletes világban”. Valójában nem tapasztalnánk soros sebességnövekedést, ha ezer MP3-feladatot indítanánk egyszerre, mert ezer processzor esetén jelentős hálózati késleltetéssel és sávszélesség-korlátokkal kellene szembenéznünk.

A legtöbb Linuxon futtatható program kis erőfeszítéssel futtatható a Beowulfon a fenti MP3-as példához hasonló módon, így számítási teljesítménye nagymértékben növelhető. A legtöbb kísérleti adathalmaz viszont – mint például az időjárás-előrejelzési adatok vagy a DNS-szakaszok – nem bonthatók fel az MP3-gyártás módján, mert az adathalmaz egyik részén végzett számítások eredménye befolyásolja az adathalmaz más részein végzendő számításokat. Az ilyen esetben végzendő számítás ahhoz hasonló, mint amikor egy óriási MP3-fájlt kell készíteni több processzor igénybevételével. Könnyű belátni, hogy ha egyetlen MP3 elkészítésének feladatát több processzorra bízzuk, a processzoroknak a munka egyeztetéséhez beszélgetniük kell egymással. Az ilyen programok erre a célra üzenetküldő programkönyvtárakat használnak, így a program által elvégzett számításról a többi processzoron futó programrész is értesül.

Két elterjedt párhuzamos programozási modell létezik: az egyik az üzenetküldő felület (MPI) programkönyvtárat használja, a másik a PVM, a párhuzamos virtuális gép. Az elvek egyszerűek, de a gyakorlatban nehéz őket hatékonyan megvalósítani. A párhuzamos programozás összetett feladat. A nagyteljesítményű számítások önmagukban is épp elég bonyolultak és kiábrándítóak, hát még, ha kódunkat drága, szeszélyes és egyedi nagyszámítógépeken kell futtatnunk! Régen a nagyteljesítményű számítások végrehajtása csak nehezen kezelhető,



szabadalmakkal védett gépeken volt elképzelhető, de ezek a megoldások szerencsére lassanként kihalnak. Egyre többen használnak a sarki számítógépboltban is beszerezhető alkatrészekből összeszerelt gépeket Linuxszal, melyekből Beowulf-géptelepet építenek. A Beowulf ár-teljesítmény aránya verhetetlen, ráadásul a nyílt és szabványos felületen a programozás szórakozásnak sem utolsó.

Mindez jól hangzik, azonban a kezdeti időkben a Beowulf – akárcsak a többi nagyteljesítményű számítási módszer – minden volt, csak nem egyszerű, leginkább a boszorkánysághoz volt hasonlatos. A Beowulf-telep létrehozása külön programok, programkönyvtárak és segédeszközök letöltését és telepítését igényelte minden egyes Linux munkaállomásra, amelyek általában vegyes hálózatra csatlakoztak. Minden Beowulf egyedi volt, akár a gépek összekötését, akár a rajtuk futó programokat nézzük, és állandó változás jellemezte őket. A géptelep kezelése és fenntartása a helyi gépek és programok mély ismeretét igényelte. Sok gond akadt akkoriban, de akárcsak a többi nyílt forrású sikertörténetnél, a közösség megtalálta a megoldást. 1994 óta a közösség és a vállalati partnerek jóvoltából a Beowulf sokat fejlődött, és megjelent a Beowulf-programterjesztések második nemzedéke. Igen, terjesztésekről van szó, amelyek CD-n jelennek meg. Nincs szükség többé az Internet

legtávolabbi zugaiból összeszedegetni a Beowulf telepítéséhez szükséges eszközöket, programokat és meghajtókat. Túl jól hangzik ahhoz, hogy igaz legyen? Olvass tovább, mert a különböző Beowulf-terjesztéseket, beszerzési helyüket és a telepítésüket is be fogom mutatni.

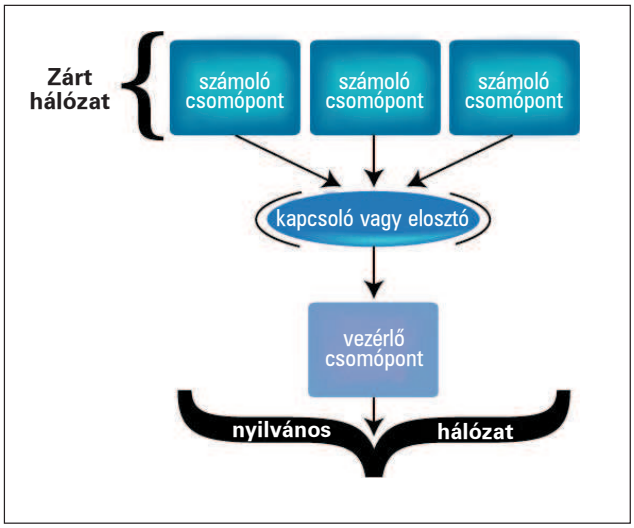
A vas szintjén minden Beowulf rendelkezik néhány olyan közös tulajdonsággal, ami megkülönbözteti néhány munkaállomás általános hálózatba kapcsolásától. A megszokott hálózattal szemben nem minden Beowulf-csomópont egyenlő, „osztálytársadalomba" rendeződnek. Minden Beowulfban egy vezérlő csomópont, és sok számoló csomópont helyezkedik el. A vezérlő csomópont a Beowulf vezérlőközpontja, ez futtatja azokat a démonokat, amelyek a számítógépek párbeszédéhez szükségesek. A vezérlő csomópont az a hely, ahonnan a rendszer agya (ez te vagy) programjaik telepítése és beállítása által kormányozza a számoló csomópontokat. A fájlrendszerek itt csatolhatók, a feladatok végrehajtása megfigyelhető, az erőforrások itt oszthatók el, és szintén itt teremthető kapcsolat a külvilággal.

A számoló csomópontok a vezérlő csomópont számítási parancsait hajtják végre, és létezésükről tájékoztatják a vezérlőt. A számoló csomópont lehet nagyon buta (kevés kódot használó), de viszonylag okos is (teljes Linux-telepítés futhat rajta). Azonban még a teljes Linux-telepítést hordozó számoló csomópontokból is hiányzik néhány olyan tulajdonság, amelyet a vezérlő csomópont nyújt, így biztosítva uralmát. Például az NFS-en keresztül a csomópontokra befűzött fájlrendszerek (mint például a felhasználók saját könyvtárai) a vezérlő csomóponton helyezkednek el. Mivel minden itt tárgyalandó Beowulf-terjesztés alapértelmezetten ezt a megközelítést alkalmazza, az egyszerűség kedvéért a cikkben felépített géptelep is ezt a megközelítést fogja utánozni. Ne feledjük azonban, hogy a valóságban bizonyos I/O-szolgáltatások az egész telepre kiterjedhetnek, a fájlok írása és olvasása több helyen is történhet, ha az adatfolyam kiegyenlítése ezt kívánja. Kezdetnek viszont talán az a legegyszerűbb, ha a vezérlő csomópontot bízzuk meg a géptelep szolgáltatásainak futtatásával és a számoló csomópontoknak szükséges adatok tárolásával.

A Beowulfban a vezérlő és a számoló csomópontok hálózatba vannak kötve. Ez a hálózat zárt, az általános hálózati forgalomtól el van különítve. A hálózati eszközök választása többnyire a felhasználó anyagi lehetőségeitől függ. A 10 Mb/s sebességű ethernettől a nagysebességű (nagyobb, mint 1 Gb/s) különleges eszközökig (pl. a Myrinet az USA-ban) széles a választék. A legolcsóbb hálózat ethernetkártyák, jelelosztók és Cat5 UTP-kábelek segítségével alakítható ki. Hacsak nem akarod a géptelep összes felhasználóját a vezérlő csomópontra telepíteni, nem árt, ha a Beowulf-vezérlőközpont egy második hálózati kártyán keresztül a külső hálózatra is csatlakozik. Ebben az elrendezésen a vezérlő csomópont a zárt Beowulf-hálózat és a munkahelyi nyilvános hálózat között az átjáró szerepét is játssza. A felhasználók a nyilvános hálózaton keresztül a vezérlő csomópontra távolról be tudnak jelentkezni, és a második hálózati kártyán át elérhetik a géptelep erőforrásait, azonban nem tudnak átlépni a vezérlőn és közvetlenül csatlakozni a számoló csomópontokhoz.

A Beowulfot a fent vázolt módon különálló számítási egységként érdemes kezelni a szervezeten belül. Ennek a megközelítésnek számos előnye van a teljesítmény, a kezelhetőség és a biztonság szempontjából. Nem csoda, hogy a jelenlegi Beowulf program ezt az elrendezést támogatja. A Beowulf fizikai kialakítása *ábránkon* figyelhető meg.

Lényegében két nézet létezik a Beowulf operációs rendszerét illetően. Hadd hangsúlyozzam, hogy mindkettő jó, bár eltérő. A két megoldás a géptelep célját illetően különböző igényeknek felel meg. Többek között a következő szempontok határozzák meg a géptelepet működtető program beállításait, valamint a hozzáférés szabályozását és ellenőrzését: a géptelep feladata, a felhasználók fajtája és száma és a futtatandó alkalmazások.



Egy jellemző Beowulf-rendszer fizikai kialakítása

Ha ezeket az elején figyelmen kívül hagyod, később bajba kerülhetsz, ezért először a két géptelep tervezésének filozófiáját tárgyaljuk.

Az eredeti Beowulfban minden csomópont teljes értékű Linuxot futtatott, és a felhasználónak az alkalmazás futtatásához minden csomóponton azonosítóval kellett rendelkeznie. Ez az elrendezés rengeteg pluszköltséget okozott minden olyan csomópontban, ahol az alkalmazásnak futnia kellett, ráadásul a helytelenül viselkedő folyamatok kezelése is nehézkes volt. Azóta ez a gépteleptípus valamelyest fejlődött. A DHCP, a Red Hat Kickstartja, az SSH, az MPI, a HTTP és a MySQL kétségkívül leegyszerűsítették a géptelep telepítésének és kezelésének feladatait, de a vezérlő csomópontra bejelentkezve a felhasználók továbbra is elérhetik a számoló csomópontokat, és a csomópontok önálló életet élhetnek.

A számoló csomópontok elérhetősége és vezérelhetősége kívánatos lehet egy adott géptelep és adott felhasználói kör esetében, ezért ez lényeges felügyeleti döntés. Két terjesztés használja ezt a modellt: az NPACI által készített Rocks és az Open Cluster Group által készített OSCAR.

A másik géptelepelméletet a Beowulf alkotója dolgozta ki. Ez a megközelítés az vezérlő-számoló viszonyt a méhkaptár mintájára valósítja meg. A vezérlő csomópont a méhkirálynő, aki teljes kromoszómakészlettel rendelkezik, tud magáról gondoskodni, és vezérli a kaptárt. A számoló csomópontok csak a kromoszómakészlet felével rendelkeznek, ők a herék, és annyi eszük sincs, mint egy marék molylepkének. A számoló csomópontokra nem lehet távolról belépni, azok egyszerűen csak az uralkodó parancsait hajtják végre. Bár a rovarpárhuzam nagyon jól hangzik, ezt az elrendezést egyszerűen SSI-nek nevezik, és az ezt megvalósító terjesztés neve Scyld. Az SSI a másik véglet a számoló csomópontok elméletében, de egyértelműen megvannak az előnyei.

Melyik modell felel meg céljaidnak? Nehéz kérdés, és végül oda lyukadunk ki, hogy attól függ, milyen jogokat szeretnél

adni a felhasználóknak a teljes rendszeren. Csak akkor hozhatsz körültekintő döntést, ha előbb eljátszol egy kicsit a különféle alapértelmezett beállításokkal.
Kezdjük az ismerkedést a különféle géptelepkezelő programokkal, a kollégáim által készített terjesztés, az NPACI Rocks telepítésével. A San Diego Supercomputer Centerben dolgozó kis munkacsoport rendkívül megbízható és könnyen használható terjesztést készített. A géptelep kiépítéséhez mindössze ábránkhoz hasonló x86-os gépek (IA32 vagy IA64) hálózata, internetkapcsolat, CD-író, CD és hajlékonylemez szükséges.
Először is látogass el a Rocks webhelyére, a ➲ http://rocks.npaci.edu címre. Rövid általános bevezetőt olvashatsz a géptelepek építéséről és a Rocks projekt sajátos módszereiről. Egy dolog azonnal világossá válik a webhely tanulmányozása során, mégpedig hogy a Rocks-terjesztés készítőinek egy cél lebegett a szemük előtt: a Beowulf géptelepeket legyen egyszerű telepíteni és kezelni! A Rocks csoport e cél elérésének érdekében

1. a terjesztést a Red Hat Linux alapján készítette el,
2. a Red Hat Linuxhoz hozzáadott olyan minden nyílt forrású programot, ami a Beowulf használatához szükséges,
3. minden géptelepkezelő programot rpm-csomagokba csomagolt,
4. az vezérlő és a számoló csomópontok telepítésének gépesítésére a Red Hat Kickstart programot használta,
5. a vezérlő csomóponton létrehozott egy MySQL-adatbázist, amely a géptelep adatait rendezi,
6. hozzáadott egy-két programot, amely az egészet egybefogja.

Csupa nagyszerű ötlet. A Rocks-terjesztés a következő telepkezelő programokat tartalmazza: PBS, Maui, SSH, MPICH, PVM, tanúsítványhitelesítő program, a Myricom általános üzenetküldő rendszere a Myrinet kártyákhoz. Amennyiben még ez sem lenne elegendő, a fejlesztők a továbbfejlesztés lehetőségét is nyitva hagyták: saját Beowulf-változatodat testreszabhatod és programjaid hozzáadásával felépítheted. Ugye, fantasztikus?
A Rocks webhelyén a bal oldali ***Getting Started*** hivatkozás elvezet egy olyan leíráshoz, amely lépésről lépésre elmagyarázza a Rocks-géptelep telepítését. A ***Step 0*** röviden leírja a géptelep fizikai kialakításának tudnivalóit. Az építőelemek másmilyenek is lehetnek, de az összeállításnak az ábránkon látható felépítésre kell emlékeztetnie.
A ***Step 1*** leírja, hogyan töltsd le a rendszerindításra képes lemezlenyomatot (ISO) a Rocks FTP-kiszolgálójáról, és miként írd fel CD-re. Az NPACI Rocks jelenleg Red Hat Linux 7.1-sen alapul, azaz két telepítő CD-je van, azonban csak az első szükséges a Rocks-géptelep telepítéséhez.
A ***Step 2*** leírja a ***kickstart*** beállítófájl elkészítését és a vezérlőcsomópont (a Rocks szóhasználatában front end) telepítését. Szerencsére a Rocks csoport webhelyén egy CGI-űrlap található, ennek kitöltésével a fájl könnyen elkészíthető. Kattints a ***Build a configuration file*** hivatkozásra, és az űrlap bekéri a ***kickstart*** fájl elkészítéséhez szükséges adatokat. Ez a fájl fogja beállítani a front end külső és belső csatolófelületeit, valamint a géptelep számára nyújtott nyilvántartó, időzítő és névkiszolgáló szolgáltatásokat. Számos tanács és alapérték segít az űrlap helyes kitöltésében.
Az űrlap kitöltése után kattints a ***Submit*** gombra. Ezután a böngésző felajánlja a fájl mentését. A fájlt ***ks.cfg*** néven mentsd, és egy DOS formátumú hajlékonylemezre másold át. Minden együtt van a Beowulf-géptelep telepítéséhez – a Rocks CD-k és a rendszeredre jellemző kickstart lemez. Kezdődhet a telepítés! A leendő front end csomópontot kapcsold be, győződj meg arról, hogy a CD-ről tud-e rendszert indítani, tedd be az első CD-t, és indítsd újra a gépet. A `boot:` parancssorba írd be a `frontend` szót, a lemezt helyezd be, és végig kísérd figyelemmel, ahogy a rendszer települ. A telepítés után a gép kiadja a CD-t, és újraindul. Az újraindulás előtt vedd ki a CD-t és a lemezt, nehogy megint elkezdődjön a telepítés. Rendszergazdaként jelentkezz be, ekkor a gép meg fog kérni az SSH-kulcsok elkészítésére. Miután a kulcsok elkészültek, futtasd az `insert-ether` parancsot. Ez a program megkönnyíti a Beowulf telepítését és kezelését, mert a számoló csomópontok adatait, amelyeket a DHCP-kérésekből nyer ki, a Rocks MySQL adatbázisába helyezi. A program menüjéből válaszd a `Compute` menüparancsot, helyezd be az első CD-t az első számoló csomópont CD-meghajtójába, kapcsold be a gépet, és várd meg, amíg a rendszer települ. A telepítés végén a gép kiadja a CD-t. Tedd át a CD-t egy másik számoló csomópontba, és kapcsold be. A fenti műveletet az összes számoló csomóponton ismételd meg.
Ennyi az egész. Az biztos, hogy sok minden egyéb zajlik a háttérben, de a Rocks készítői ezeket a folyamatokat szándékosan elrejtették előled. Ha a érdekelnek részletek, a folyamatosan bővülő leírás a webhelyen fellelhető.
Gondolom, már alig várod, hogy Beowulfodon alkalmazásokat futtass. A webhelyen a ***Step 4*** pontban rövid ismertető olvasható az MPI-t, illetve a PBS-t használó alkalmazások futtatásáról. Még egy tanács: az MPI-feladatokat az `mpi-launch` parancs a Myrineten keresztül, míg az `mpi-run` parancs az etherneten keresztül indítja el.
A cikk itt véget ér, azonban nemsokára újra jelentkezem egy másik könnyen telepíthető Beowulf-terjesztés ismertetésével, azután egy harmadikkal és még eggyel. Azt követően számítórácsba kötjük őket. Végül jöhet a világuralom.

Linux Journal május, 97. szám

Glen Otero
Ph.D. fokozatot szerzett immunológiából és mikrobiológiából, továbbá a Linux Prophet nevű tanácsadó céget vezeti a kaliforniai San Diegóban.

Kapcsolódó címek

Beowulf.org ➲ http://www.beowulf.org
Kickstart ➲ http://www.redhat.com/docs/manuals/linux/RHL-7.2-Manual/custom-guide/ch-kickstart2.html
Linux Clustering Information Center ➲ http://www.lcic.org
Maui ➲ http://www.supercluster.org/maui
MPI ➲ http://www-unix.mcs.anl.gov/mpi/index.html
Myrinet ➲ http://www.myri.com
OSCAR ➲ http://oscar.sourceforge.net
PBS ➲ http://www.openpbs.org
PVM ➲ http://www.csm.ornl.gov/pvm
Red Hat Linux ➲ http://www.redhat.com
Rocks ➲ http://rocks.npaci.edu
RPM ➲ http://www.rpm.org
Scyld ➲ http://www.scyld.com

Grafikus meghajtók Linux alatt

Robin számba veszi a különböző nyílt forrású és kereskedelmi grafikus kiszolgálókat, kártyákat és meghajtókat.

A Linux alatti grafikus teljesítményt a grafikus alkatrészek, a videomeghajtó és a grafikus csatoló is befolyásolja. Az alábbiakban sorra vesszük a nyílt forrású Xfree86 kiszolgálót, a kereskedelmi Xi kiszolgálót, az ATI Fire GL munkaállomásokba szánt videokártyáját, az nVidia játékosokat megcélzó GeForce3 Titanium kártyáját és a Wacom digitábla-felület linuxos meghajtóit.

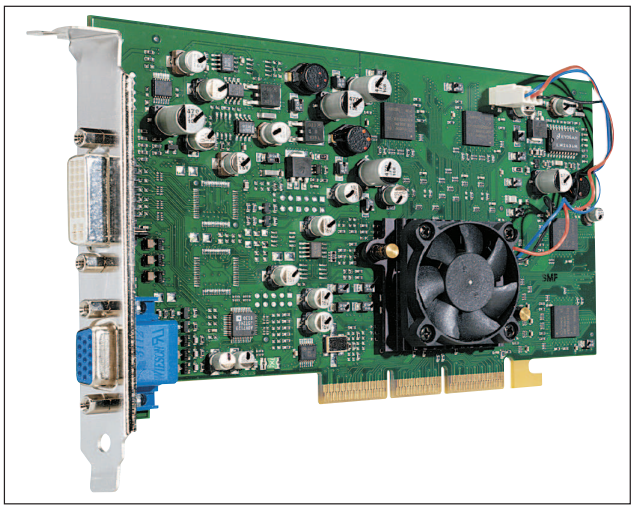
Az Xfree86 a legismertebb X-kiszolgáló, hiszen szinte valamennyi Linux-terjesztés tartalmazza. A legtöbb otthoni felhasználó számára ez az a módszer, amely az X Window rendszer grafikus alapjaiért felelős. „Az Xfree86 4.2-es, 2002. januárban megjelent kiadása élsimított betűtípusokat, hibajavításokat, és számos új meghajtót is tartalmaz többek között Radeon 8500 és Matrox G550 kártyákhoz" – mondja ***David Dawes***, az Xfree86 munkacsoport elnöke és a kiadásért felelős igazgatója. Dawes testületi tag és vezető programmérnök a Tungsten Graphics cégnél, amelyet egykori VA Linux-alkalmazottak alapítottak abból a célból, hogy meghajtóprogramokat, OpenGL API-kiterjesztéseket és DRI-fejlesztéseket végezzenek. A Luxi méretezhető betűi – mind a TrueType, mind a Type 1 – részét képezik a 4.2-es kiadásnak. A Bigelow & Holmes Rt. ajánlotta fel ezeket az új betűtípusokat, amelyek eredetileg Ikarus digitális formátumban készültek, majd az URW++ Design and Development GmbH végezte el a TrueType és Type1 átalakításokat.

Az Xfree86 4.2.0 részlegesen már megvalósítja az új X leképező (rendering) kiterjesztésmódszerét. A legtöbb alkatrész által támogatta egyszerű összeadó műveletek segítségével a leképező képes élsimított szövegek és geometriai objektumok létrehozására, valamint áttetsző képátfedések kivitelezésére. Még megvalósításra várnak a geometriai alapműveletek és a képek hasonló transzformációja. Csupán három alkalmazást alakítottak át úgy, hogy élsimított szöveget jelenítsen meg a 4.2.0-s Xfree86 alatt: `xterm`, `xditview` és `x11perf`.

A 4.2.0 kiadás újításai a Darwin Mac OS X-re is hatással voltak. A Mac OS X alatt egy új, rendszergazda nélküli mód gondoskodik arról, hogy az X-ügyfélprogramok az Aqua asztal felületén ablakokat jeleníthessenek meg. Az XDarwin most már támogatja a Xinert, vagyis képes rá, hogy az ablakokat két képernyőnyi terjedelemben elnyújtsa.

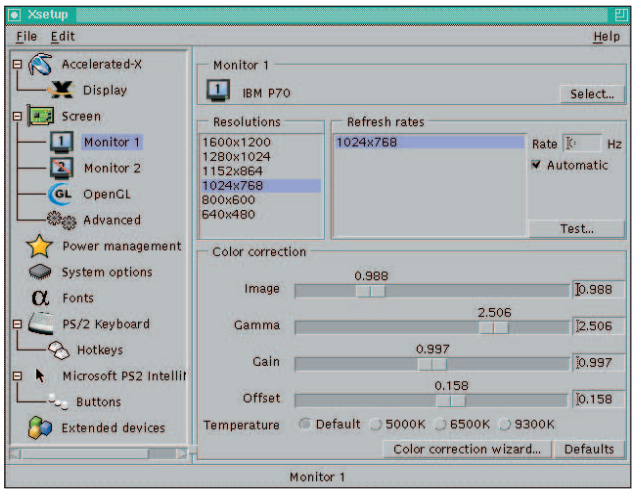
Amikor megkérdeztük Dawest, hogy mi a legfőbb vágya az Xfree86 következő kiadásával kapcsolatban, a könnyebb telepíthetőség volt a válasz. „Folyamatosan fejlődünk ezen a területen. A 4.3-as változatban én magam is dolgozni fogok rajta."

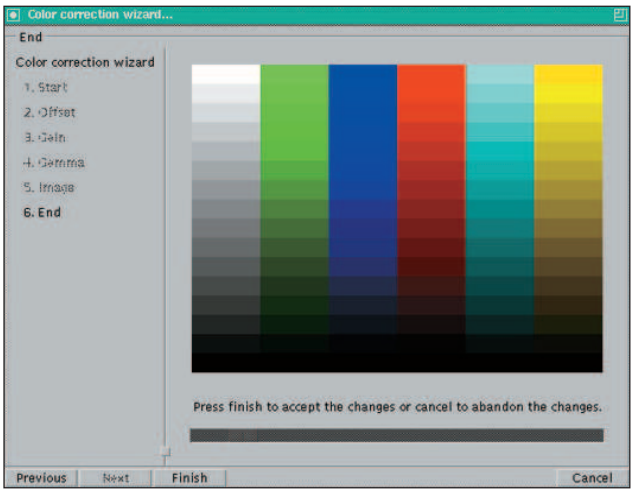
Az olyan kereskedelmi kiszolgálótermékek, mint a Metro-X és a Xi Graphics választási lehetőséget kínálnak az Xfree86-tal szemben. A HP és az IBM is készít saját X-kiszolgálókat, amelyeket munkaállomásaikkal együtt hoznak forgalomba. A Metro-X létrehozója, a Metro Link a videofelvevő tévé set-top készülékek piaca felé mozdult el. „A linuxos Metro-X fejlesztését leállítottuk, bár még hozzáférhető" – mondja ***Morgan Von Esson***, a CEO munkatársa. „Jelen pillanatban a 25 ezer dolláros PVR SDK piac áll a figyelmünk középpontjában." A Metro Link és az ATI technológia partnereként működik az ATI set-top HDTV hivatkozási felület fejlesztésében. A Metro Link jelentős mértékben hozzájárult az Xfree86 4.x fejlődéséhez, mivel felajánlotta hozzá saját futásidejű betöltőprogramját. Ezzel az Xfree86 képes rá, hogy dinamikusan töltsön be meghajtókat, még olyan operációs rendszerek alatt is, amelyek a dinamikusan csatolt függvénykönyvtárakat egyébként nem támogatják.



1. kép ATI Fire GL 8700

Az Xi Graphics 2001 novemberében adta ki először az Accelerated-X Summit X nevű kiszolgálóját, amely egyre több grafikus kártyát támogat – jelenleg több mint harminc kártyát és hordozható számítógépet. Az Accelerated-X kiszolgálónak asztali gépekhez (DX), hordozható gépekhez (LX), többfejes (MX) és munkaállomásokhoz (WX) készült változatai érhetők el. „Ami termékünket vonzóvá teszi, az a korlátlan idejű terméktámogatás, a teljesítmény és a megbízhatóság"– mondja ***Dave Methvin***. „Támogatjuk a szabad program elgondolást is, és a Solaris-típusú hordozható gépeinkhez készült PCMCIA-hibajavításokat például nyílt forrásban tettük hozzáférhetővé. Az Xfree86 azonban nem nyújt olyan tesztelési és támogatási rendszert, amelyet mi kínálunk kereskedelmi termékeinkhez." ***Thomas Roell*** – jelenleg a XiG CTO munkatársa – még egyetemistakorában, Németországban ültette át az eredeti X Konzorcium-féle megvalósítást az Intel x86-os rendszerre, majd ingyen az MIT rendelkezésére bocsátotta (a későbbi Xfree86). Az Accelerated-X szolgáltatási között találjuk a Color Magic színkezelő rendszert, a DualView-t, ami két monitort képes kezelni, a Video for Windowst a nagy teljesítményű YUV képernyők támogatására, a Stereo 3D-t a szemüvegek kezeléséhez, és a Power Throttle szolgáltatást, amely a hordozható gépek takarékos áramfelvételéért felelős. Az Accelerated-X vissza tudja fogni az APCI-t használó processzorok áramfelvételét, sőt, a grafikus alkatrészek egyes részegységeit ki is tudja



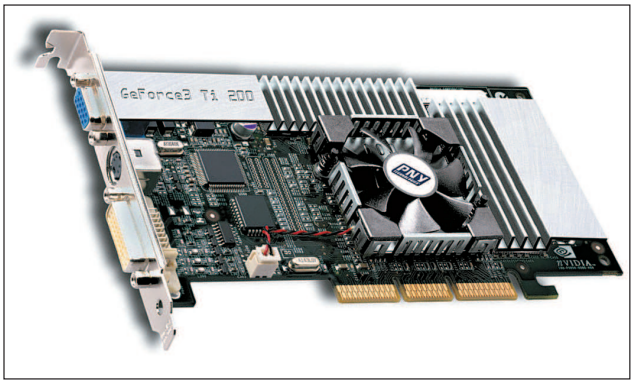


2. kép Az XiG Xsetup beállítóeszköz a Color Correction varázsló segítségével lehetővé teszi a színhőmérséklet gyors módosítását

kapcsolni, ha éppen nincs rájuk szükség. Methvin szerint ezzel a megoldással az Xfree86-tal szemben a hordozható gépek tápja akár két órával is tovább bírhatja.
Az OpenGL-gyorsításhoz az XiG nem a DRI-t, hanem egy kisebb API-t, az XDA-t használja. „A DRI túlságosan nagy és bonyolult" – mondja Methvin. „Volt némi nehézségünk az Xfree86 felépítésével kapcsolatban." Az Accelerated-X telepítése kihívást jelenthet a már Xfree86-ot használóknak, mivel a két rendszer nem igazán szeret együtt élni. Az Accelerated-X csoport saját, hatékony OpenGL-megvalósítást készített, és azt javasolják, hogy az ütközések elkerülése érdekében rendszerünkből távolítsuk el a Mesa-támogatást. Az Accelerated-X-et használó OpenGL-fejlesztőknek az Xfree86-megfelelő Mesa fejlesztőcsomag helyett az XiG-féle OpenGL fejlesztőkészlet letöltését és használatát javasolják.
Az XiG egy ingyenesen letölthető, időkorlátos próbaváltozatot kínál. Sőt, a próbaváltozatot először mindenképpen le kell töltenünk, és csak utána vásárolhatjuk meg. Amíg a vásárláskor kapott kulccsal nem aktiváljuk az Accelerated-X programot, addig az minden indítás után csak 25 percig fog működni (az újraindítások száma azonban nem korlátozott). A DX-változat ára 39 és 99, az LX változaté 69 és 139, az MX-é 129 és 249, míg a WX változat ára 129 és 379 dollár között mozog.
Mi a 99 dolláros DX RADEON Platinum változatot telepítettük. A leírás kicsit zavaros volt, főleg azért, mivel nem voltunk biztosak abban, mit is kell tennünk a rendszerünkön lévő Xfree86-tal és a Mesával.
Az Accelerated-X telepítésének lépései:

1. A grafikus felületen történő bejelentkezést kapcsoljuk ki, és állítsuk le az éppen futó X-kiszolgálót.
2. Adjuk ki a `su root` utasítást.
3. A Mesát távolítsuk el vagy kapcsoljuk ki a rendszerünkből.
4. Távolítsuk el vagy kapcsoljuk ki az `agpgart` rendszermagmodult.
5. Telepítsük a Linux-rendszermag forrását.
6. Telepítsük az X-szolgáltatásokért felelős rpm-modult (XSVC).
7. Adjuk ki a `make xsvc` utasítást.
8. Futtassuk a `make xsvctest` parancsot.
9. Amennyiben szükséges, az MTRR-, AGPGART- és SMP-támogatást a megfelelő beállításokkal fordítsuk a rendszermagba.
10. Telepítsük a Summit RPM-et.
11. Futtassuk az Xsetup programot, ezzel grafikus kártyánkat, monitorunkat és egerünket szöveges módban állíthatjuk be.
12. Futtassuk a `startx`-et.
13. Futtassuk újból az Xsetup programot, hogy a további beállításokat már grafikus felület alatt végezhessük.

Néhány nem várt akadályba ütköztünk, mivel Red Hat helyett Debian Linux alatt végeztük a telepítést. Az `alien` program segítségével az összes rpm-állományt deb formátumba alakítottuk át. Az `xsvctext` program tudtunkra adta, hogy az



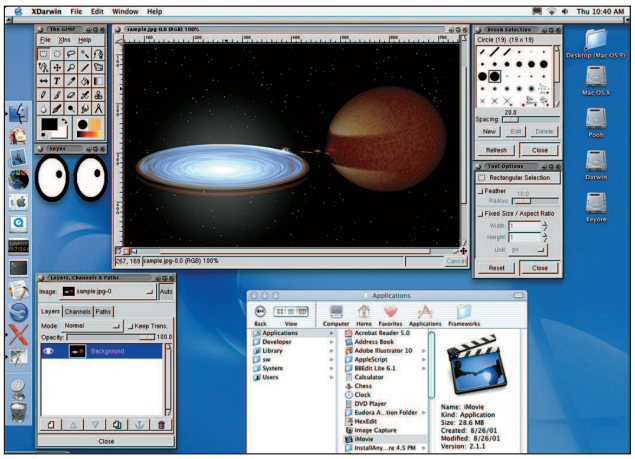
3. kép PNY Verto nVidia GeForce3 Ti200

Accelerated-X a nagy teljesítményű ASUS A7A266-típusú alaplapunkat nem támogatta, és ez azt jelentette, hogy az AGP-gyorsításról le kellett mondanunk. A deb formátumba való átalakítás miatt az `xsvc modutil alias`-okat kézzel kellett létrehoznunk, akárcsak a ***.xinitrc*** és ***.xserverrc*** állományokat.

```
# vi /etc/modutils/aliases
.
.
alias char-major-10-175 agpgart
alias char-major-10-179 xsvc
.
.

tbird:/etc/modutils# update-modules
```

A `gpm`-t le kellett állítanunk, és az `Xsetup` programban a MS Intellimouse egeret választottuk ki, mert optikai Logitech MouseMan Wheel egerünket csak így tudtuk működésre bírni. Az XiG legfőbb piaci jelenlétét az Accelerated-X adja. „A nagyteljesítményű munkaállomásokba szánt kártyák irányába mozdulunk el” – mondja Methvin. „Csodálattal szemléljük az új kártyák teljesítményét. Nemrég készítettünk meghajtót a 3DLabs Wildcat II 5110 kártyához.” Az XiG-féle OpenGL-támogatás 1.2.1-es változatát éppen most váltja fel az 1.3-as.



4. kép Az Xfree86 és a MacGimp egy Mac OS X Finder ablakkal együtt. A két különböző ablakozórendszert itt együtt láthatjuk működni

Az XiG nem támogatja az nVidia kártyáit. „Szeretnénk támogatást nyújtani az nVidia kártyákhoz – mondja Methvin –, de az nVidia nem adja ki kártyái leírásait. Meg kell elégednünk azzal, hogy a versenytárs ATI termékekhez fejlesztünk mind gyorsabb meghajtókat.” Az X Window rendszerrel kapcsolatosan Methvin legnagyobb bánata az, hogy az asztali felületek terén túl sokféle lehetőség létezik. „A Gnome olyan módon működik együtt az Xfree86-tal, amely más X-kiszolgálók működését megakadályozni látszik, és a különböző ablakkezelőkből is túl sok fajta létezik. Úgy vélem, egy szabványos grafikus munkafelület hiánya az, ami a visszatartja az olyan nagy fejlesztőket, mint például az Adobe abban, hogy alkalmazásaikat Linuxra is átültessék.”

A grafikus kártyagyártó ATI 2001 áprilisában felvásárolta a Fire GL Graphics céget, és vele együtt munkaállomásokba szánt kártyáinak teljes termékvonalát is. Ezeket a kártyákat (például Fire GL) a 3D-animátoroknak és mérnököknek szánják, nem az otthoni felhasználóknak. Míg az otthoni játékos a Quake futtatásához egy olyan nagy teljesítményű kártyát használ, mint az ATI RADEON, addig a játék tervezője még egy annál is nagyobb teljesítményű FireGL kártyára épít, amikor a játék 3D-s világát létrehozza, vagy mozifilmekhez tervez különleges hatásokat (effects). E kártyák között a fő különbség az OpenGL-gyorsításban megmutatkozó teljesítményeltérésben van. A FireGL javára írható még, hogy több ablakban is nagy képfrissítést biztosít, támogatja a kétszeres gyorstárazást és a két monitor kezelését, valamint nagyobb megbízhatóságot és jobb terméktámogatást nyújt.

Az ATI érintetlenül hagyta azt a német céget, amely a Fire GL-hez készít linuxos meghajtókat. A FireGL-meghajtó zárt kódú, és valószínűleg az is marad. „A szellemi tulajdonnal és a piaci versennyel kapcsolatos gondok is felmerülnek olyankor, amikor zárt kódú meghajtót szeretnénk nyílt forrásúvá tenni – mondja ***Ed Huang***, a munkaállomás részlegvezetője. „Úgy hiszem, mi rendelkezünk a leggyorsabb Linux alatti OpenGL-megvalósítással, és nem akarjuk kiadni a versenytársaknak.”

A FireGL 2 és a FireGL 4 ugyanazt a zárt forrású meghajtót használja. A Radeon esetében azonban más az ATI stratégiája, hiszen meghajtója nyílt forrású. A csúcsteljesítményű FireGL 4 mintegy 1500 dollárba kerül (forrás: ➲ http://buy.com), míg az olcsóbb Fire GL-ért körülbelül 725 dollárt kérnek el (forrás: ➲ http://cdw.com). Ezeket a kártyákat két új Fire GL-modell váltja fel. A Fire GL 8800 a felvásárlás óta az első teljes egészében ATI fejlesztette Fire GL kártya, valamint az első olyan, ami a Radeon lapkára épül. A 128 MB-os kártya a legtöbb alkalmazás esetében körülbelül ötven százalékkal nyújt majd jobb teljesítményt, mint a Fire GL 4, és várható ára kevesebb mint 900 dollár. A Fire GL2 helyére a Radeon 8800LE lapkán alapuló 64 MB-os Fire GL 8700 lép. Mintegy ötven százalékkal jobb a teljesítménye, mint a GL2-nek, és várhatóan 400 dollárnál is kevesebbe kerül majd.

5. kép
A Wacom Cintiq egy 15” LCD képernyőt és egy vezeték nélküli tollat kapcsol össze, amely a nyomásérzékenység 512 szintjével rendelkezik. A Linuxot használó animátorok közvetlenül a képernyőre rajzolhatnak vele

Valamennyi FireGL-kártya ugyanazt a linuxos meghajtót használja. A FireGL-meghajtó (jelenleg 4.1.0-s Xfree86 és 6.2-es libc a követelménye) az ATI weboldaláról érhető el ***.tgz*** és rpm formátumban. Mérete körülbelül 5,5 MB. A grafikus lapkagyártó nVidia új, nagy teljesítményű játékokhoz szánt kártyáját, a Geforce3 Titaniumot PC-ibe a Compaq, a Dell, a HP és az IBM is beépíti. Az Apple asztali gépeiben és a Microsoft Xboxaiban is ez a kártya található. Az nVidia jó évet zárt, értékpapírjai szerepeltek a legjobban a 2001-es S&P 500-on.

A GeForce3-modellek közé tartozik a Ti200 (160 dollár körül), és a még nagyobb teljesítményű Ti 500 (300 dollár körül). Ezek a kártyák gyors, nagy felbontású élsimítást szolgáltatnak, amiért a GPU a felelős (HRAA Quincunx), valamint DVI- és tévé s-video kimeneteket is tartalmaznak. A vizuális termékek létrehozóinak az nVidia még az utóbbiaknál is nagyobb teljesítményű Quadro2-Pro kártyáját kínálja (körülbelül 615 dollárért).

A Linuxot használóknak az nVidia kártyáiban az egységesített meghajtófelépítés lehet érdekes. „Ugyanazt az egységes kódot használjuk valamennyi kártyánk linuxos és windowsos meghajtóihoz – mondja ***Dwight Diercks***, a programfejlesztési rész-

leg elnökhelyettese. „Ha új kártyát készítünk, a linuxos meghajtók már készen állnak, akárcsak a windowsos meghajtók. Egy új grafikus lapka elkészítése utáni 60 napon belül mind a windowsos, mind linuxos meghajtókat is kiadjuk hozzá."
Az OpenGL-megvalósítást az összes felületre kiadták. „Az egységes meghajtószerkezet teszi lehetővé, hogy valamennyi kártyánkhoz egyetlen bináris állományt használjunk – mondja ***Nick Triantos***, az OpenGL és a Linux munkaállomások részlegének vezetője. „Ha a meghajtó teljesítményén javítunk, az valamennyi termékünkre jótékony hatással lesz. Más cégek, amelyek különböző meghajtóprogramokkal dolgoznak, nem győznek lépést tartani a frissítésekkel."
Diercks szerint az nVidia igyekszik annyit újrahasznosítani a már megírt meghajtókódokból, amennyit csak lehet.
„A legfőbb különbséget a két rendszer meghajtóiban a Linux alatti OpenGL GLX-réteg, és a Windows alatti Wiggle-réteg jelenti. Az összes tökéletesítés, új szolgáltatás és bővítés kevés többletmunkával Linux alá is bekerül." Az Nvdriver linuxos rendszermagmódú meghajtó. A windowsos meghajtó a VXD miniport meghajtó.
A linuxos Nvdriver zárt forrású meghajtó. Létezik nyílt kódú nVidia-meghajtó is, ami az Xfree86 része. Ez képes 2D-s műveletek megvalósítására (gyorsított videó-, DVD-lejátszás, monitorfelismerés), az OpenGL 3D-s gyorsítás elveszett, amikor az Xfree86 felépítése a 4.x-változatokban megváltozott.
Az nVidiánál a ***Mark Voikavich*** vezetette tíz fejlesztőmérnökből álló csapat foglalkozik a Linux-támogatás megvalósításával. Bár a Microsoft saját DirectX protokollja nyilván nem része a linuxos meghajtóknak, de az összes egyéb szolgáltatás Linux alatt is elérhető. „A két monitor kezelését lehetővé tevő TwinView Linux alatt is megtalálható" – mondja Diercks. „A linuxos hordozható gépek támogatják a többfejes technológiát abban az esetben, ha több kijelzőt szeretnénk használni – például előadások tartásakor." A felhasználó az ablakokat átvonszolhatja az egyes képernyők között. A linuxos meghajtó valamivel kisebb, mint a windowsos, mivel az OpenGL-ben és a DirectX-ben is megvalósított szolgáltatásokat nem kell ismét tartalmaznia. A beállításoktól függően a linuxos meghajtó gyorsabb is lehet, mint windowsos társa.
Az nVidia szorosan együttműködik az Xfree86 csoporttal. „A jövőben még több szolgáltatást igyekszünk nyújtani a hordozható gépekhez, például a gyorsbillentyűk használatát, amelyek dinamikusan mozgatják az ablakokat az egyes kijelzők között – mondja Triantos. „X alatt nincsen olyan szolgáltatás, amely egy ablakot csak egyetlen képernyőn jelenít meg." Szintén hiányzik az X alól az a szolgáltatás, amellyel a gyorsított ablakokban megjelenő képet lehet menteni (screen capture). Windows alatt létezik a meghajtóhoz, de Linux alatt ugyanehhez a művelethez az adott alkalmazás segítségét kell kérni, például egy OpenGL-es `glReadPixels` képernyőmentéshez Quake alatt az F11 billentyű lenyomásával. Mivel a gyorsított ablakok egy üres területet hoznak létre az asztalon – amire aztán írhatnak – nehezebb művelet menteni őket, mint egy egyenletesen fekete hátterű ablakot.
Az nVidia nemrégiben adta ki Personal Cinema nevű grafikus kártyáját, mellyel az ATI hasonló AIW Radeon kártyájával kívánja felvenni a versenyt. „A Personal Cinema kártya videók szerkesztését és rögzítését teszi lehetővé, de ezeket a szolgáltatásokat a linuxos meghajtó nem tartalmazza" – mondja Diercks. „Minket érdekelne ennek a megvalósítása, de nem vagyunk meggyőződve róla, hogy elegendően nagy rá az igény Linux alatt." Az ATI AIW Radeon viszont rendelkezik egy új Video4Linux meghajtóval a videofilmek rögzítéséhez.
Az nVidia az új lapkák kifejlesztésekor erősen támaszkodik a Linuxra. „Red Hat Linux 6.2-t használunk egy 1500 gépből álló kiszolgálótelepen"– mondja Diercks. „Mérnökeink ezen a rendszeren a lapkatervezés folyamatát ellenőrzik."
Nem minden grafikus linuxos meghajtó áll a videózás szolgálatában. A Wacom digitábla-felületek a Gimp-felhasználók vagy a linuxos munkaállomásokon dolgozó mozifilmanimátorok körében népszerűek. ***Frederic Lepied***, az Xfree86 egyik fejlesztője felelős a linuxos Wacom-meghajtókért. „Az előző munkakörömben egy Solaris X-kiszolgálóhoz kellett hozzáférnünk digitábla-felületen keresztül, és a Wacom IV eszközökhöz szükséges protokollok a Wacom weboldalán elérhetők voltak" – mondja Lepied. „Később a solarisos Wacom-meghajtót átültettem a linuxos Xfree86 alá. Úgy váltam Xfree86-fejlesztővé, hogy az Xinput-kiterjesztést Xfree86 alatt is munkára tudtam bírni."
A linuxos Wacom-meghajtó fejlesztése 1995-ben kezdődött. Lepied az egyetlen fejlesztője, de munkájához sokan hozzájárulnak. A fejlesztőmunka folyamatos erőfeszítés azért, hogy a meghajtó az egyre újabb modelleket is támogassa. Lepied munkáját a Wacom is segíti, de anyagi hozzájárulást a munkaadójától, a MandrakeSofttól kap. Lepied a Mandrake Linux-terjesztés egyik fejlesztője és csoportigazgatója. A nyílt forrású linuxos Wacom-meghajtó a Wacom IV, Wacom V és az USB-protokollokat támogatja.

Linux Journal április, 96. szám

Robin Rowe
a MovieEditor.com internetes és televíziós videoalkalmazásokat készítő cég egyik partnere. Írásai a Dr. Dobb's Journalban, a C++ Reportban, a C/C++ Users Journalban, a Data Based Advisorban jelentek meg és számos tanácskozás anyagában megtalálhatók. A Robin által készített programok sorában található többek közt az a kiszolgálóalapú videoszerkesztő rendszer, amit a Manhattan 24 órás televíziós hírcsatorna, a Time Warner New York One, illetve a kapcsolódó honlap ➲ http://www.ny1.com/ is használ. Elérhető a robin.rowe@ movieeditor.com címen.

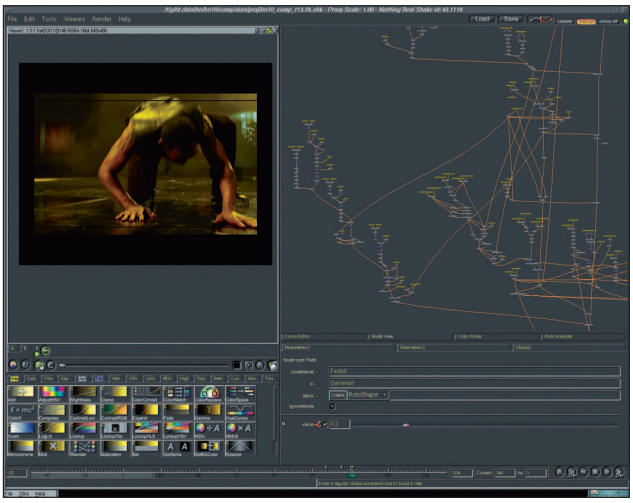
Kapcsolódó címek

ATI ➲ http://www.ati.com
Bigelow & Holmes (nem hivatalos honlap)
➲ http://www.myfonts.com/FontFoundry78.html
DRI ➲ http://dri.sourceforge.net
Mesa ➲ http://mesa3d.sourceforge.net
Metro Link ➲ http://www.metrolink.com
nVidia ➲ http://www.nvidia.com
PNY ➲ http://www.pny.com
Quake ➲ http://linuxquake.com
Wacom ➲ http://www.wacom.com
Wacom-meghajtók ➲ http://www.people.mandrakesoft.com/~flepied/projects/wacom
XDarwin ➲ http://www.xdarwin.org
XFree86 ➲ http://www.xfree86.org
Xi Graphics ➲ http://www.xig.com
XiG OpenGL-Devkit ➲ http://ftp.xig.com/pub/3Daccelx

A Tippett Studio és a Nothing Real's Shake

Robin vizsgálódásainak középpontjában a Tippett Studio által különleges hatások megalkotására használt és Linuxon futtatott Nothing Real's Shake áll.

A videó-összeszerkesztő programokat a filmstúdiók arra használják, hogy a különleges hatásokat (effects) vagy animációkat filmjeleneteké kapcsolják össze. Ezek a programok ugyanarra képesek a mozgóképpel, amire a Gimp és a Photoshop az állóképek esetében. Napjaink legelterjedtebb csúcsminőségű szerkesztőprogramjának a Nothing Real's Shake csomagja tűnik. A program futtatható Linuxon, Windowson és IRIX-en, az Apple pedig nemrég erősítette meg a híresztelést a Nothing Real's Shake megszerzésével kapcsolatban.
A Tippett Studio olyan rémisztő hatásokra szakosodott, mint rovarok és egyéb teremtmények megalkotása. Vizsgáljuk meg a kaliforniai Berkeley-ben működő stúdió munkáját és a módszert, ahogyan a Shake-et Linuxon alkalmazzák.
A Tippett Studio hatásaival több akadémiai díjat is nyert, például A Jedi visszatér és a Jurassic Park című játékfilmekben végzett munkáért, és jelöltként szerepelt a Starship Troopers, a Dragonheart, a Willow, a Dragonslayer és Hollow Man című filmekben való közreműködéséért. Ottjártamkor a 2002 márciusában bemutatásra kerülő Blade II című vámpírfilmhez készülő hatások munkálatai folytak. „A Shake-Linux párost az Evolution óta, körülbelül másfél éve használjuk" – tájékoztat ***Alan Boucek*** szerkesztésvezető. „A Shake-et egy kétprocesszoros Athlon-rendszeren futtatom, és az a legizgalmasabb benne, hogy rendkívül gyors. A munka nagy részét a saját gépemen anélkül el tudom végezni, hogy a leképezőtelephez kellene fordulnom"– e ponton hozzáteszi még, hogy a szerkesztőgárda volt a Stúdió első csapata, amelyik Linuxra váltott. „Amikor a Shake-et üzembe helyeztük, a következő feladat a linuxos feladatfolyam létrehozása volt. A háromdimenziós leképezések nagy részéhez a Pixar Rendermanját használjuk. A szürke leképezések közvetlenül a Mayából származnak, az animátorok anyagaikat azonban egy egyszerű Renderman-folyamaton engedik át a napi áttekintés céljából" – magyarázza Boucek.
A Shake faszerkezetű felületet használ a hatások kezelésére. Minden bemenet csomópontokon halad át, amelyek szűrőket vagy függvényeket képviselnek a kép végső formájának elérési útján. Az összetett hatások leképezése hosszabb időt vesz igénybe – az idő pedig a stúdiók egyik legféltettebb kincse.
A különleges hatások szakértői majdnem valós időben, gyakran a végleges felbontás negyedrészével dolgoznak a Shake segítségével egy munkaállomáson, és a folyamat végén készítik el a végleges, teljes felbontású változat létrehozására a leképezőtelepen futtatandó kötegelt parancsállományt.
Két évvel ezelőtt, amikor a Tippettnek új gépek vásárlására volt szüksége, 5000 dolláros SGI x86-os PC-k megvétele mellett döntöttek a 24 000 dolláros SGI Mips Octane gépek mellé. „Először SGI PC gépeket vettünk, de miután gyártásuk leállt, saját gépeinket elkezdtük az alapoktól felépíteni. A gépekkel kapcsolatban sok pénzt megtakarítottunk, ennek jelentős részét azonban a saját magunknak nyújtott technikai támogatás emésztette fel." A PC-k és SGI számítógépek mellett szép számmal találhatók Macintoshok is. A PC-k egy részén nem Linux, hanem a Shake és a háromdimenziós Maya modellezőcsomag windowsos változata fut. Boucek a Linux mellett teszi le voksát:
„Az NT-t nem tudjuk összehozni a leképezőteleppel. Amikor éjszaka hazamegyünk, a linuxos gépek tovább dolgoznak. Az NT munkaállomások nem bírják a leképezőtelep kétszeres teljesítményét. Még mindig sok SGI-gépünk van, éppen most fejeztük be a Linuxra való áttérést."
„Már az hatalmas előny, hogy az eredeti kód írójával beszélhetünk" – mondja ***Christian Rice***, a műszaki igazgató.

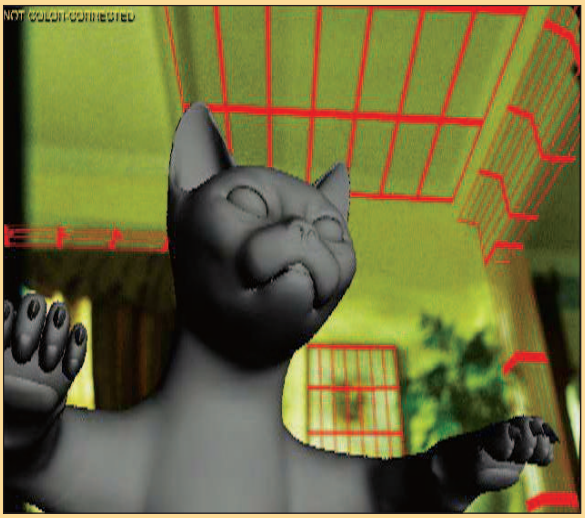


A Shake használata a Blade II-ben egy Wesley Snipes-jelenet összeállításakor. Az ablakkezelő az MWM, amely a használat folytonosságának érzését biztosítja az SGI IRIX-ről való átállás során

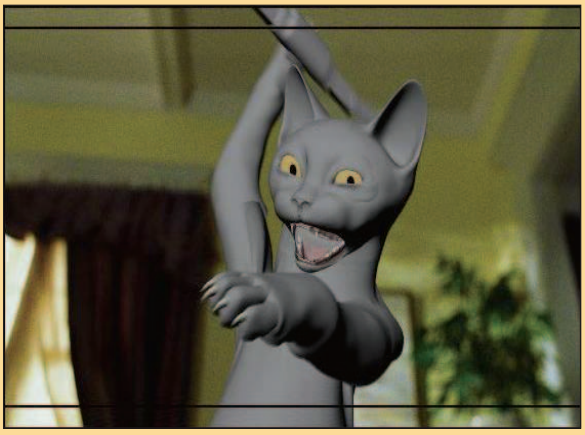
A Linux alkalmazásával hivatalos kötöttségek nélküli, személyes párbeszédre nyílik lehetőségünk az alkotóval. Számos akadályt sikerült már ily módon legyőznünk. Jóllehet, mindez nagyobb felelősséggel és bizonytalansággal jár együtt. Ha a Világhálón nem bukkanunk megoldásra, az még nem jelenti azt, hogy veszett ügyünk van. A fejlesztővel közvetlen kapcsolatot felvéve gyakran nyílnak meg előttünk olyan különleges lehetőségek, amelyekről addig nem is álmodtunk, nem is beszélve azokról a további szolgáltatásokról, amiket kifejezetten igényeink kedvéért valósítanak meg.
„A 150 fős csapatban tíz programozó dolgozik" – tájékoztat Boucek. „Két- és háromdimenziós megvalósításokon, valamint a csővezeték-támogatáson munkálkodnak. Sokat foglalkoznak olyan rendszerek együttműködésének megteremtésével, amelyek korábban erre nem voltak felkészítve". Programozóink körülbelül 25 kétdimenziós és 75 háromdimenziós munkát végző grafikust tevékenységét támogatják. Bár a Shake számos beépített hatással rendelkezik, a stúdiók gyakran továbbiakat készíttetnek, hogy ezzel saját egyéni képvilágukat erősítsék. Miként a Gimp és a Photoshop, a Shake is lehetőséget nyújt kiegészítők írására C, illetve C++ nyelven.

Az orosz macska lelövésének folyamata a 153-as képen

1. kép



2. kép



3. kép

4. kép

5. kép

1. kép A jelenet nyers CG-modellje. A hátteret a szerkesztő cseréli ki a kamera által felvett képekre

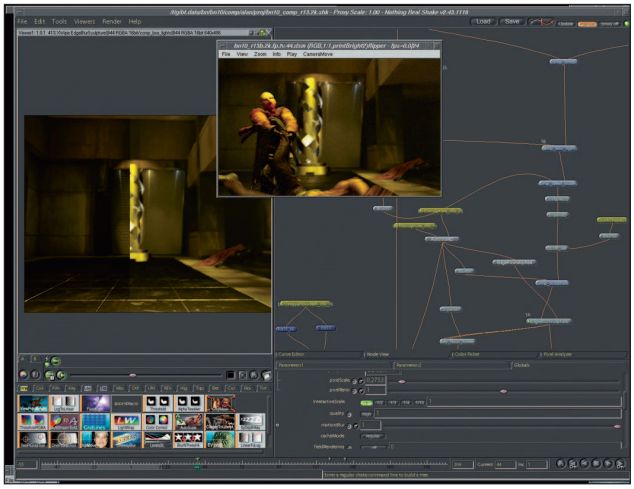
2. kép A grafikus elkészíti a beállított mozgásnak megfelelő mozdulatokat

3. kép A nyers macskamodell elhelyezése a számítógépes grafikai környezetben és a háttérre történő ráfektetése

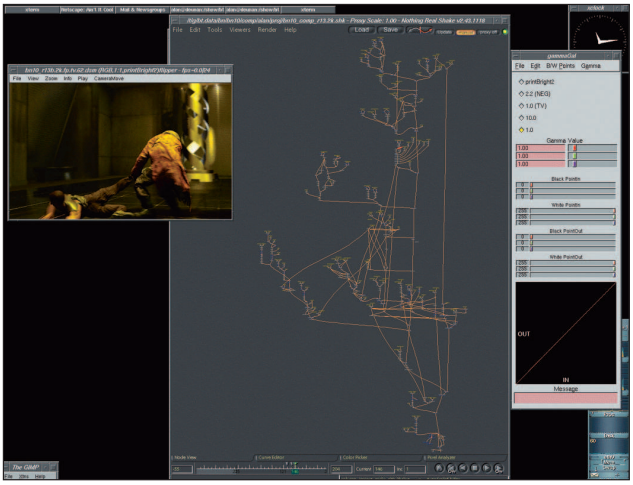
4. kép A macskamodell finomítása és mozgatása a háttér előtt

5. kép A Shake a végső képet összeszerkeszti a már színhelyes és hibajavított hátérrel

A film negatívját a Warner Bros Production biztosította.



Shake-előnézet a bal oldalon a megosztott képernyőn. A Flipper flipbook-lejátszó a jobb oldalon. A bal alsó sarokban a saját kiegészítések ikonjai láthatók



A Flipper a bal oldalon látható, jobbra pedig a GammaGal. A Shake folyamatábrája, amelynek segítségével a részek egyetlen jelenetté állnak össze

„Eleinte igen nehéz feladat kiegészítőt írni a Shake-hez” – figyelmeztet ***Qin „Jean” Shen***. Nincs teljes fejlesztői leírás, ezért kezdetben a fejlesztői készletből átmásoltam egy egyszerű példát, majd mialatt a kiegészítésen dolgoztam, elektronikus levélben kérdéseket intéztem a Nothing Realhez. Válaszaikban igen segítőkésznek mutatkoztak.” Shen elmagyarázza, hogy a Nothing Real ugyan próbálkozott jól használható leírás elkészítésével, de a kiegészítők annyira egyediek, hogy nehéz minden részletre kitérni. A kiegészítők fejlesztőinek még arra is figyelmet kell fordítaniuk, hogy minek kell a memóriában lennie ahhoz, hogy a Shake önműködően kiolvashassa őket onnan. „A Shake vonalas leképező – jegyzi meg Boucek – nem olvassa be a teljes képet mindjárt a folyamat elején. Ez teszi olyan gyorssá.”

Shen cáfolja azt a vélekedést, hogy a tanulmányait befejező programozó ne kaphatna izgalmas munkát egy filmstúdió programfejlesztői részlegében. Bár kapott ajánlatot az ILM tájáról is, ő a Tippett mellett döntött, mert szerinte egy kisebb stúdió nagyobb lehetőségeket teremt. „Amikor 1999-ben dolgozni kezdtem, még az Alias/Wavefront Composert használtuk. Kiegészítéseket írtam hozzá, aztán a Shake-hez is, miután átálltunk”. Shen jelenleg két belső eszköz, a Flipper (flipbook-lejátszó) és a GammaGal (monitorbeállító eszköz) fejlesztésén dolgozik. „A Flipper és a GammaGuy Motif és IrisGL használatára íródott. Átültettem OpenGL-re és FLTK-ra, így lett a GammaGuy-ból GammaGal.”

„A Photoshop fontos tulajdonsága a gammaszint-beállítás” – magyarázza ***Darby Johnston*** programfejlesztő. „A GammaGal valami hasonlót tesz lehetővé nagy sebességgel IRIX és Linux alatt.” Mivel a filmeknek nagyobb a dinamikatartománya, mint a monitoroknak, a felhasználóknak folyamatosan szabályozniuk kell a szintet, hogy a sötét és világos tartományok részleteit láthatóvá tegyék. A filmen dolgozó grafikusok úgy játszanak a gammaszintekkel, mint a két dimenzióban dolgozók a nagyítással.

„Enyhe túlzásnak tűnik az IRIX-eszközök Linuxra való átalakításáról átültetésként beszélni” – mondja Johnston. „Az esetek többségében csupán fordításról van szó.” Johnston munkája főleg hagyományos, szálak és megosztott memóriahasználat nélküli C-kódolást jelent. Johnston olyan csapat tagja, amely egy egységes programcsomaghoz ír összekötő kódot és parancsfájlokat.

„Még mindig léteznek a Linuxszal kapcsolatos növekedési gondok” – jegyzi meg Boucek. „Sok linuxos fejlesztés olyan, mintha a Microsofttal küzdene, pedig nekünk megbízható felületre van szükségünk, amely eszközeinket futtatni tudja.” Amikor a Tippett a Linuxszal kezdett foglalkozni, az nVidia-meghajtók még próbaállapotban voltak és nem működtek megfelelően. „Néha még egy glLines-vonalat sem tudtam jól megrajzolni – mondja Shen –, inkább a glStripet használtam helyette – rengeteg ilyen kódunk van.” Boucek még hozzáteszi, hogy „a korai hónapok számos hibája az nVidia-meghajtókra volt visszavezethető, ezenkívül a Maya még mindig elég megbízhatatlan Linuxon, ha nVidia GeForce2 vagy Quadro2 grafikus kártyát használunk.” A gondok ellenére a Linuxra történő áttérésben az nVidia-meghajtóknak kulcsszerepük volt.

„Az eredeti nVidia-meghajtók teljesítménye erős ösztönzést jelentett a Linuxra való átálláshoz, bár elmondhatjuk, hogy a Quake irányába mutató javítási törekvések túlmutatnak a két dimenzión” – mondja Johnston.

Nemrégiben próbáltuk ki a XiG-et egy ATI Radeon 7500-as kártyával egy 1,4 GHz-es Athlon processzorral, és a 2D teljesítménye sokat javult. 1920×1200-as felbontáson az nVidia éppen hogy elérte a 30 fps-t, míg a Xig meghajtóprogramok 40 feletti eredményt mutattak fel. Hozzá kell tennünk, hogy három dimenzióban az nVidia-meghajtók észrevehetően gyorsabbak voltak a Mayában.

„Keményen dolgozunk azon, hogy mindenkinek az asztalán Linux működjön, a Maya megbízhatósági gondjai miatt azonban még nem értük el a célunkat” – mondja Boucek. Szeretné, ha a Linux és meghajtói a nagyobb megbízhatóság és teljesítmény felé mozdulnának el.

A Square a másik olyan filmstúdió, amely elképesztő munkát végzett a Shake segítségével a Final Fantasy című film megalkotása során. „Négy éve az IRIX-szel kezdtük” – emlékszik vissza ***James Rogers***, a látványhatások igazgatója. „Linuxos leképezőtelepeink a gyártás közepén lettek munkába állítva.” A Square meglévő IRIX-gépeit használja munkaállomásként, ezután az adatok a szokásos leképezéskezelőn keresztül jutnak a Shake-hez, amely a linuxos leképezőtelepen fejezi be a folyamatot. „A Linuxot végül a munkaállomásokon is elkezdtük kipróbálni” – mondja Rogers. (A Square nemrégiben jelentette

be, hogy felhagy a filmgyártással, és hawaii-i irodáit 2002. március 29-én bezárta.)
A Shake telepítéséhez három fájlt (55 MB) kell a Nothing Real honlapjáról letöltenünk:
a licenckezelőt (***lmutil.Z***, 184 k),
a programot (***shake-linux-v2.46.0116.tar.bz2***, 28 MB) és
az oktatót (***shake-tutorial-v2.46.0116.tar.gz***, 26 MB).

```
gunzip lmutil.Z
tar xvfj shake-linux-v2.46.0116.tar.bz2
tar tvfz z/shake-tutorial-v2.46.0116.tar.gz
tar xvfz z/shake-tutorial-v2.46.0116.tar.gz
./lmutil lmhostid
```

A kétszeres ellenőrzés érdekében a tar `tell` kapcsolóját szoktuk használni, hogy a kibontás előtt lássuk, mit fog tenni. Az `lmutil` program egy 12 számjegyű azonosítót ír ki ezt elküldve megkapjuk kéthetes próbakódunkat (***key.dat***) a Nothing Realhez. A kapott kódot másoljuk a megadott könyvtárba, indítsuk el az X-et, és futtassuk a Shake-et:

```
cp key.dat shake-v2.46.0116/keys/
startx
shake-v2.46.0116/bin/shake
```

Ha korábban még nem találkoztunk a Shake-kel, két hét nem túl idő sok ennek az összetett eszköznek a megismerésére. Írásom idején a Shake jövője még nem teljesen tisztázott.
Az Apple egy 2002. február 6-án kiadott rövid közleményében megerősítette a Nothing Real megszerzéséről terjedő híreszteléseket. Boucek a Tippett Studiótól megjegyzi:
„Most olyan leképezőmaggal rendelkeznek, amely mindenki másénál gyorsabb. Ez igen értékes lehet a QuickTime-ban vagy a Final Cut Próban. A Shake számunkra a Linuxon annyira értékes, hogy aggódni kezdtünk amiatt, hogy csak az OS X rendszeren lesz elérhető. Az viszont nem ártana, ha olcsóbb lenne, és több munkahelyet engedhetnénk meg magunknak. Általában érdeklődünk az OS X iránt, hiszen sok macintoshos gépünk is van."
A BDS-alapú OS X az Apple felülete, mely támogatja az NFS-t, és további Unix-tulajdonságok is segítik abban, hogy a filmgyárak munkafolyamataiba való beillesztése ne okozzon nehézséget.
Az Apple a vásárlási híreket megerősítő kétmondatos nyilatkozatában csak ennyit állított: „tervezzük a Nothing Real eljárásainak saját termékeinkben történő jövőbeni felhasználását." 2001-ben a Nothing Real bejelentette, hogy az Apple OS X rendszerén is elérhetővé teszi a Linux-, Windows- és IRIX-rendszereken már futtatható Shake-et. Mindenesetre az Apple még nem kötelezte el magát a Shake OS X-re történő megjelentetése mellett.
Mivel az Apple vásárlás után eddig még sosem folytatott fejlesztést más operációs rendszerekre, könnyen lehet, hogy a Shake a jelenlegi állapota egyúttal a Linuxon elérhető utolsó változat. Habár a Shake leképezőtelep kiszolgálóprogramja talán továbbra is támogatni fogja a Linuxot, erre már láthattunk példát az Apple Linux Darwin QuickTime kiszolgálójánál. Ha a Shake GUI jövőbeni változatai csak Macintosh gépeken fognak futni, a gépbeszerzéssel járó többletköltség erős akadályt jelenthet a stúdiók számára, igaz, az Apple a Shake árának csökkentésével ellensúlyozhatja ezt. A munkahelyenkénti tízezer dolláros ár mellett az Apple bedobhat egy kétezer dolláros iMacet, akár ingyen is.

Néhány stúdióigazgatót elbizonytalanít az Apple hallgatása a Shake-kel kapcsolatos terveit illetően, mások azonban, mint Boucek, derűlátásuknak adnak hangot, és bíznak benne, hogy az Apple jó ajánlattal fog nyitni a filmgyártás piaca felé.
Mivel a Linux-piac nagy részét a kiszolgálók képezik, a Linuxszal foglalkozó cégek inkább ezzel a területtel törődnek. Annak ellenére, hogy ez tervként nem fogalmazódott meg, a munkaállomások terén a Red Hat gyakorlatilag szabványnyá vált. Néhány felhasználó szerint a kiszolgálókra összpontosító Red Hat nem a legjobb választás asztali gépeken való használatra, ***Robert Young***, a Red Hat igazgatója mégis elismeri, hogy a linuxos közönség több ízben lehurrogta, amikor kijelentette, hogy a Linux soha nem lesz népszerű az asztali gépeken.
Az Apple OS X felől érkező kihívás a Linux fejlesztőit arra ösztönözheti, hogy még inkább felhasználóbarát felületet nyújtsanak. Az OS X-nek köszönhetően ***Ernest Prabhakar***, az Apple termékmenedzsere 2002 februárjában a USENIX BSDCON rendezvényen bejelenthette, hogy a BSD asztali gépeken háromszor népszerűbb a Linuxnál.
Az Apple felrázhatja a mozgófilmhatások piacát is. ***Steve Jobbs*** immár mind az Apple-nek, mind a Pixarnak az igazgatója. Az is előfordulhat azonban, hogy a feltörekvő versenytárs, a Silicon Grail RAYZ programja lesz a Nothing Real felvásárlásának nagy győztese. Számos stúdió a kialakult bizonytalan helyzet miatt szemet vetett a RAYZ-ra. A következő hónapban a RAYZ Linuxon történő alkalmazását vizsgáljuk meg közelebbről.

Linux Journal május, 97. szám

Robin Rowe
a MovieEditor.com internetes és televíziós videoalkalmazásokat készítő cég egyik partnere. Írásai a Dr. Dobb's Journalban, a C++ Reportban, a C/C++ Users Journalban, a Data Based Advisorban jelentek meg és számos tanácskozás anyagában megtalálhatók. A Robin által készített programok sorában található többek közt az a kiszolgálóalapú videoszerkesztő rendszer, amit a Manhattan 24 órás televíziós hírcsatorna, a Time Warner New York One, illetve a kapcsolódó honlap ➲ http://www.ny1.com/ is használ. Elérhető a robin.rowe@ movieeditor.com címen.

Kapcsolódó címek

Apple ➲ http://www.apple.com
Blade II ➲ http://www.blade2.com
Darwin QuickTime
➲ http://www.opensource.apple.com/projects/streaming
Final Fantasy ➲ http://www.finalfantasy.com
FLTK ➲ http://www.fltk.org
Maya ➲ http://www.aliaswavefront.com
Nothing Real ➲ http://www.nothingreal.com
nVidia ➲ http://www.nvidia.com
OpenGL ➲ http://www.opengl.org
Pixar RenderMan ➲ http://www.pixar.com/renderman
Silicon Grail ➲ http://www.sgrail.com
Tippett Studio ➲ http://www.tippett.com
USENIX BSDCON
➲ http://www.usenix.org/events/bsdcon02

Digitális mozgóképsorok saját kezűleg (2. rész)

MEncoder – kikövezett út a hatékony tárolás felé, ismét megújult formában.

Legutóbbi írásom végén azt ígértem, hogy ebben a részben a gyakorlaté lesz a főszerep, ami természetesen igaz is, azonban mielőtt belevetnénk magunkat a sűrűjébe, meg kell vizsgálnunk az előző cikkben leírtakat. Ennek az az oka, hogy a cikk megírása óta az MPlayer (vele együtt a MEncoder is) látványos változásokon ment keresztül, ami rendkívül kedvező fordulat, ugyanis rengeteg új lehetőség és javítás került bele, ám ezzel együtt némiképpen a kezelés módja is megváltozott.
Nem kell megijednünk, hiszen az alapelgondolás maradt a régi, csupán az egyes kapcsolók és értékek megadásának módja változott néhol, feltehetően a könnyebb vagy inkább strukturáltabb használat és felépítés érdekében.
A fejlesztés tehát gőzerővel folytatódik, újabb és újabb kiadások jelennek meg (a cikk írásának pillanatában a 0.9pre2 változatnál tartunk, ám elképzelhető, hogy holnapra már ez is „elavult” lesz). Ez a „gőzerő” a csomag statisztikáin meg is látszik. Az egyik legnagyobb linuxos weboldalon, a Freshmeaten (➲ http://www.freshmeat.net) jó néhány kimutatás található, köztük a legnépszerűbb projektek húszas toplistája is, amelyen az első helyet a rendszermag foglalja el 100 százalékponttal, ezt követi az MPlayer 95,5 százalékponttal (összehasonlításképpen: a 3. helyezett cdrecord csak 47 százalékpontot szerzett). Azt hiszem, a számok önmagukért beszélnek, a program igen kedvelt, így hát mi is tanulmányozzuk tovább!

Néhány szó a legújabb fejlesztésekről

A cikk terjedelméből kifolyólag nem lehet minden egyes újítást, javítást, kényelmi megoldást felsorolni, néhányat azonban ki szeretnék emelni közülük.
Az egyik az operációs rendszerektől független egyedülálló feliratozás. Nagyszerűségét abban látom, hogy míg ablakos nézetben az árnyékolt feliratot vagy órát a filmre helyezi, addig teljes képernyős módban a képernyőn lévő holttérre próbálja meg helyezni. A holttér a monitor (4:3) és a film (16:9; 2,25:1) képarányának eltéréséből adódik.
Léteznek olyan feliratozók is, amelyek a feliratot ablakban képesek megjeleníteni, ám növelik az ablak méretét (4:3 képarány), hatalmas, kétszerakkora helyet foglalva el a képernyőből. Ha valakinek mégis ez tetszik, az Expand nevű szűrővel ablakban is készíthet magának holtteret. A másik vonzó javítás, hogy a lejátszómotor és a kodekek fejlesztésének köszönhetően a lejátszási sebesség is javult. A lassúbb masinával rendelkező felhasználók ezt leginkább számításigényes DVD-filmek lejátszása során érzékelhetik.
A legutóbbi cikk óta kialakítottak egy videoszűrő réteget (ez az egyik oka annak, hogy a paraméterezés megváltozott), amely ugyanazt a munkát végzi feltételezhetően ugyanazon a módon, ám sokat egyszerűsít a kezelésen és annak megértésén, egyben felkészülés is a további képkezelő szűrők fogadására. Az új változat több ilyennel is bővült, ezek közül az egyik az előző írásom végén hiányolt képszegélylevágás (cropping) (a folyamat fordítottja is létezik, amit az Expand szűrő valósít meg). A régi videoszűrő eljárások (flip, postprocess) természetesen a programban maradtak, csak ezúttal a `-vop` kapcsoló után felsorolva lehet megadni őket. Az új változatban már a `libavcodec` kodekcsalád DivX-tagjával is lehetőség nyílik a kétmenetes kódolás alkalmazására. Ezekkel a megoldásokkal gyakorlatilag egy profi DVD-„átalakító”, DivX-es AVI-készítő eszköz áll rendelkezésünkre.

Vissza a MEncoderhez

Kezdjük rögtön a változások ismertetésével! Javaslatom, hogy mindenekelőtt a 0.9 pre2 változatot telepítsük (34. CD/Magazin/MPlayer könyvtár), hogy az itt leírtak ne térjenek el attól, amit olvasóink hiba nélkül alkalmazni tudnak.
A telepítés folyamata teljesen megegyezik az MPlayerről szóló cikkünkben leírtakkal (Linuxvilág 2002. március, 28. oldal). Azt csiripelik a kismadarak, hogy hamarosan megjelenik a pre3 változat, amelyet már előrefordított csomagok formájában is el lehet érni, így a hosszas telepítési, fordítási folyamatot megtakaríthatjuk magunknak.
A legfontosabb változás az egységes videoszűrő réteg bevezetése (a továbbiak az MPlayerre is igazak, a lejátszás szintén ily módon zajlik). Ha a MEncoder kimeneti képtulajdonságait szeretnénk megadni, a `-vop` kapcsoló mögött kell őket felsorolni, egymástól vesszővel elválasztva. A sorrend nem közömbös! A program a kapcsolólistában szereplő szűrőket fordított sorrendben, a folyamatban hátulról visszafelé fűzi láncba. Rossz sorrend esetén előfordulhat, hogy nem azt kapjuk, amit szeretnénk, néha a MEncoder nem is tud mit kezdeni a hibás értékekkel, és futása hibával megszakad. Ilyen például, amikor a kép egy adott részét szeretnénk kivágni és más nagyságúra átméretezni. Ebben az esetben, ha a kivágást előbb adjuk meg – ami azt jelenti, hogy a folyamat először az átméretezéssel fog kezdeni –, meglehet, hogy az átméretezett képből már nem onnan és nem ugyanakkora darabot vág ki a képszegélylevágás művelet, mint amit mi eredetileg szerettünk volna. A képszegélylevágás művelet a másik örömmel fogadott újítás. Sokszor a bemeneti eszközről (ez leggyakrabban a DVD-lemez) nyert képnek van egy vékony fekete kerete, sőt, az is előfordul, hogy a 16:9 arányú kép 4:3 dobozban jelenik meg, és a maradék terület fekete. A holttér tárolása nemcsak helyveszteséget okoz, de a lejátszásnál többlet-processzoridőt igényel, azonkívül nem is esztétikus. A program eme lehetőségét a `-vop crop=<ØrtØkek>` kapcsolóval használhatjuk. A képszegélylevágás utáni értékek helyére először a kivágás szélessége, magassága, ezután a kivágás vízszintes, majd függőleges kezdőhelye kerül, egymástól kettőspontokkal elválasztva. A kimeneti kép átméretezését a korábbi `-x, -y` kapcsolók helyett szintén a `-vop` kapcsoló után felsorolandó `scale=<x:y>` értékekkel adhatjuk meg, ahol az `x` és `y` értéke a kép szélességének és magasságának felel meg. Ezek a leggyakrabban használt szűrők, de bőven akadnak mellettük mások is. Ezekről az MPlayer leírásának idevágó szakaszában olvashatunk részletesebben.

Kódoljunk DivX-et!

Ez a feladat már korántsem olyan látványos, mint a lejátszás. Kép helyett csak állapotjelző értékeket látunk, ráadásul rendkívül lassú. Az eljárás egyes gépeken a lejátszási sebességnél talán egy picivel gyorsabban is működik, ám a legtöbb esetben kétmenetes kódolást alkalmazunk, ami a tömörítési időt a kétszeresére növeli. Erre a jobb minőség elérése érdekében van szükség. Mint már említettem, a videotömörítés alapja az, hogy az egymás után következő képek csak kis mértéken különböznek egymástól. Ezt kihasználva egyenletes mozgásnál szép eredményeket érhetünk el. Ezzel csak az a baj, hogy a mozgás általában nem egyenletes, ezért a gyors mozgások esetén a kép eltorzulhat, és romlik a minősége. Úgy védekezhetünk ellene, ha változó bitrátájú tömörítést alkalmazunk, amely a gyors mozgásoknál kevesebb veszteséggel dolgozik, így azonos képminőséget mutat fel, mint a lassúbb mozgások. Ehhez azonban tudnunk kell, hol vannak a gyors mozgások. A másik gond a tömörített méret kiszámítása: ha ugyanis változó tömörítési mértéket alkalmazunk, a kimeneti fájl méretét még csak becsülni sem tudjuk. A kétmenetes tömörítési eljárás mindkét nehézséget megoldja, ez első lépésben állandó bitrátával tömöríti a filmet (ezt természetesen nem használjuk), közben elemzi a mozgásokat és naplózza őket. A következő lépésben újra elejéről kezdi a tömörítést, és az előző lépés során készített naplófájl alapján gyakorlatilag az egyes pillanatokban alkalmazott tömörítési arányt úgy „csoportosítja át" , hogy a lassúbb mozgásoktól „elveszi", és a gyorsabbaknak „adja", így végeredményben ugyanakkora méretű, ám a gyors mozgásoknál is jó képminőséget biztosító eljáráshoz jutunk.
A MEncoder hárommenetes tömörítésre is képes, amely egy olyan kétmenetes eljárás, amelyben a hangsáv külön lépésben kerül az AVI-ba. Ez az első lépés. Előnye, hogy mivel a hangsáv mérete és hossza ismert (feltételezzük, hogy a hang éppolyan hosszú, mint a film), így a program ennek a lépésnek a végén a kívánt adathordozó méretéthez (650, 700, 800 megabájt) viszonyított képsáv tömörítésének mértékét ajánlja fel számunkra (több CD esetén a kapott értéket szoroznunk kell). Ezután következhet a másik két lépés.

DVD-ből AVI

Előfordulhat, hogy szerényebb méretben másolatot kell készítenünk egy filmről. Ha MEncodert használunk, erre nagyon egyszerűen lehetőségünk nyílik. A lejátszáshoz hasonlatos módon, 1–2 további kapcsolóval és némi processzoridővel (10–12 óra, esetleg több) segíthetünk magunkon. Mik a teendők? Először is a filmet vissza kell nyernünk az adathordozóról, amit a program elvégez helyettünk. A következő lépés a keret levágása. Ehhez tudnunk kell a film fizikai méretét (PAL-szabvány esetén 720×576), amelyet lejátszáskor az MPlayer (a MEncoder szintén) a konzolra ír. Ezek után az MPlayer segítségével meg kell határoznunk, hogy mekkora részt és honnan kell kivágni a képből. Ezt úgy tehetjük meg, hogy mentünk (capture) egy képet, majd szerkesztőprogram segítségével keretet húzunk arra a részre, amit látni szeretnénk, ezt követően megnézzük az értékeit. A másik lehetőség: az MPlayer segítségével a látni kívánt részt nézegetési próbákkal határoljuk be.
Ezek után át kell méreteznünk a képet, hiszen a legtöbb esetben a fizikai felbontás nem a helyes képarányt tükrözi (lásd Linuxvilág 2002. április, 48. oldal), és sokszor a DVD-n alkalmazottnál kisebb felbontással is megelégszünk. Ezt a `-vop scale=<szØlessØg>:<magassÆg>` kapcsolóval adhatjuk meg. Az értékeket az eredeti képarány alapján számológép segítségével célszerű kiszámolni. Ezek után meg kell adnunk a kép és a hang formátumát, majd kétszer (háromszor) le kell futtatni a tömörítést.
Vegyünk egy példát, amelyben hárommenetes kódolást alkalmazva megtartjuk az eredeti ac3 hangot, és tömörítésre a divx4 kodeket használjuk 1400 kbit/s tömörítési aránnyal. Az eredeti anyag 720×576-os, melynek a kép szélein 8 pixel, a tetején és az alján pedig 128 pixel széles fekete keret található (ebben az esetben a látszólagos és a fizikai képarány megegyezett).
A kimeneti kép 640×272-es felbontású lesz. A fenti eredmény eléréséhez az alábbi parancsokat kell kiadni.

1.
```
mencoder -dvd 2 -ovc frameno
➥-oac copy -o frameno.avi
//a kimeneti fÆjl neve
k telezı!
```

Ennek hatására létrejön egy kép nélküli AVI, amely a film hangját tartalmazza, az alkalmazandó tömörítési arányt (1400 kbit/s) pedig eredményül a futás végén kapjuk meg.

2.
```
mencoder -dvd 2 -pass 1 -vop
➥scale=640:272,crop=712:320
➥:8:128 -oac copy -ovc divx4
➥-divx4opts br=1400 -o
➥mymovie.avi
//A program tudja, hogy a
hang a frameno.avi fÆjlban
talÆlhat .
```

Egy AVI-t (ez most lényegtelen) és egy ***divx2pass.log*** nevű fájlt kapunk, amely a mozgáselemzés eredményét tartalmazza. Ez az, ami fontos.

3.
```
mencoder -dvd 2 -pass 2 -vop
➥scale=640:272,crop=712:320
➥:8:128 -oac copy -ovc divx4
➥-divx4opts br=1400 -o
➥mymovie.avi
```

Ez az utolsó lépés. A program mind a ***frameno.avi***, mind a ***divx2pass.log*** fájlokat érzékeli, és ezek alapján elkészíti a végleges filmet.
Az itt felsorolt értékeket előre kiszámoltam, sajnos ezek minden lemez esetében eltérőek, mindig újra kell őket számolni, majd célszerű legalább a videoszűrőkre vonatkozókat MPlayerrel ellenőrizni. Még oldalakon át mesélhetnék a program további képességeiről, lehetőségeiről és az alkalmazható huncutságokról, ám azt hiszem, egy fél magazin is kevés lenne, ha mindent le szeretnék írni, épp ezért minden érdeklődőnek figyelmébe ajánlom a magyar nyelvű leírást, melynek tanulmányozásával és rengeteg „gyakorlással" tapasztalatokat szerezhetnek a videotömörítés nemcsak linuxos, de általános fortélyairól is, miközben a programban rejlő számtalan apró lehetőséget is elsajátíthatják. Az MPlayer törtetlen fejlődését és egyre nagyobb népszerűségét elnézve, azt hiszem, lesz még miről írnom a továbbiakban.

Komáromi Zoltán (komi_@freemail.hu) 21 éves, a BME hallgatója, mellette PHP-programozóként dolgozik. Kedvenc területe a multimédia. Kedveli a nagy társaságot, az érdekes embereket, a jó filmeket és mindent, ami mozgalmas. Szabadidejében röplabdázik.

Kapcsolódó címek

- http://www.MPlayerHQ.hu/homepage/
- http://www.mplayerhq.hu/DOCS/Hungarian/documentation.html

Adatbázis-kapcsolat a Perl Sybase modul segítségével

Valóshelyzeti megoldások: Andrew megmutatja, hogyan építsük fel a Perl DBD::Sybase modulját a TDS könyvtárakkal.

Számos rendszergazdának – beleértve magamat is, ha parancsnyelvekről van szó, a Perlre esik a választása. A Perl a manapság használt szinte összes operációs rendszeren jelen van, köszönhetően annak, hogy a forráskódja szabad, és fejlesztők százai dolgoznak folyamatosan azon, hogy kibővítsék a képességeit. A Perl-modulok között egészen figyelemre méltó ***Tim Bounce*** DBI-je (Database Independent, azaz adatbázis-független). A Perl DBI egy olyan API-t biztosít, amelyen keresztül bármilyen adatbázishoz kapcsolódhatunk, legalábbis azokhoz, amelyekhez létezik megfelelő DBD-modul (a DBD ez esetben Database Dependentet jelent, vagyis adatbázisfüggőt). Ha vetünk egy pillantást a Perl moduljaira a ➲ http://cpan.org-on vagy a Perl HQ-n, egész sor adatbázis-modult találhatunk: az Informixhez, az Oracle-hoz, a Sybase-hez és az IBM DB2-jéhez stb. is létezik modul – meglepő azonban, hogy a Microsoft SQL-hez való kapcsolódáshoz egyetlen DBD-modult nem lelünk. Létezik azonban egy DBD::ODBC-modul, azonban ennek használatához egy külön eszközkezelőre és megfelelő eszközökre lenne szükségünk. De ahogy rögvest kiderül, egy másik mód is létezik.

A Sybase és a Microsoft közötti megállapodásnak köszönhetően a Microsoft SQL-kiszolgálóját ugyanazzal a hálózati protokollal látták el, mint a Sybase adatbázis-kiszolgálóját, nevezetesen a TDS-sel (Tabular Data Stream). Az SQL 7.0-ig a Microsoft a Sybase ügyfelét hivatalosan támogatta, mindazonáltal a Perl DBD::Sybase-modulja, melyet a Sybase szabadon letölthető ügyfélkönyvtáraival fordítottak, egészen az SQL 2000-ig minden Microsoft SQL-hez képes kapcsolódni.

A TDS 8.0 bemutatásával – mely támogatja az SQL 2000 által használt TDS 7.0-t – a TDS 4.2-vel történő együttműködés megszakadt. Mindemellett a DBD::Sybase-modul fordítható a ➲ http://freetds.org-ról letölthető TDS könyvtárakkal, melyek támogatják a TDS 7.0-s változatát, így az SQL 2000-hez is képesek kapcsolódni. A következőkben áttekintjük, milyen módon is teszik ezt.

Az ügyfelet a Red Hat Linux 7.1-en és 7.2-n próbáltam ki. Mivel e projekt során forráskódokkal dolgozunk, szükségünk lesz egy fordítóra, például a GCC-re. Ugyanígy a Perl- és DBI-csomagok is könnyedén feltelepíthetők a Red Hat Linux telepítő CD-iről, melyeken rpm-formátumban találhatók meg.

Első lépésben töltsd le a ***DBD-Sybase-0.94.tar.gz***-t egy neked tetsző helyre a számítógépeden. Ezt a fájlt megtalálod a ➲ http://www.cpan.org-on, illetve a szerző honlapján is a ➲ http://www.mbay.net/~mpeppler címen. A FreeTDS forráskódját a ➲ http://www.freetds.org címről töltheted le. A szükséges állomány neve ***freetds-0.53.tgz***, ez a cikkírás pillanatában a legfrissebb változat. Erősen ajánlom neked az ezen oldalakon található leírások elolvasását is!

Ezt követően a `gunzip` és `tar` parancsok segítségével csomagold ki a letöltött fájlokat, majd lépj be az újonnan létrejött ***freetds-0.53*** könyvtárba, és add ki a `./configure -with-tdsver=7.0` parancsot. Ennek következtében egy olyan `Makefile` jön létre, mely a számítógépednek megfelelő `freetds`-t fog létrehozni, és egyúttal alapértelmezett protokollként jelöli ki a TDS 7.0-t.

Most futtasd a `make`-et, majd a `make install` parancs kiadásával telepítsd fel az éppen lefordult `freetds`-t. Alapértelmezésben ez a parancs a keletkezett fájlokat a ***/usr/local/freetds***-be másolja, melyen a `configure script --prefix=`*`k nyvtÆrnØv`* kapcsolójának segítségével változtathatunk. Ehhez a művelethez rendszergazdai jogosultságokkal kell rendelkezned:

```
make install
```

A parancs kimenete:

```
Making install in include
make[1]: Entering directory
         `/home/atrice/freetds-0.53/include·
make[2]: Entering directory
         `/home/atrice/freetds-0.53/include·
```

…és így tovább. A legutolsó üzenetek valahogy így fognak kinézni:

```
if [ -f /usr/local/freetds/etc/freetds.conf ]; \
then :; \
else \
/usr/bin/install -c -m 644 freetds.conf
➥/usr/local/freetds/etc/freetds.conf;
\
fi
make[2]: Leaving directory
➥`/home/atrice/freetds-0.53·
make[1]: Leaving directory
➥`/home/atrice/freetds-0.53·
```

A `freetds` fordítása és telepítése itt fejeződött be.

Ezt követően a DBD::Sybase fordítását kezdjük meg, mely az előzőhöz nagyon hasonlóan zajlik. Elsőként a legfontosabb a `SYBASE` környezeti változót a bash használatával a `freetds` elérési útjára beállítani:

```
export SYBASE=/usr/local/freetds
```

Ellenőrizzük a változó megfelelő beállítását:

```
echo $SYBASE
/usr/local/freetds
```

Ezt követően lépjünk a ***DBS-Sybase*** könyvtárba, és futtassuk a `Makefile.PL` állományt:

```
perl Makefile.PL
Sybase OpenClient not found.
```

Trevor Price parancsfájlja, mellyel a Northwind nevű példaadatbázist kérdezhetjük le

```
#!/usr/bin/perl

#
# A db2 dbi meghajt tesztje
#

use DBI ;
$user = ·sa· ;
$passwd = ·password· ;


$dbh = DBI->connect(·DBI:Sybase:server=file1·,
$user, $passwd);
$dbh->do("use Northwind");

$action = $dbh->prepare("sp_help") ;
$action->execute ;
$rows = $action->rows ;
print "rows is $rows\n";

while ( @first = $action->fetchrow_array ) {
         foreach $field ( @first ) {
         print "$field\t";
         }
         print "\n";
}

exit(0);
```

```
The DBD::Sybase module needs access to a
➥Sybase server
  to run the tests.
To clear an entry please enter ·undef·
Sybase server to use (default: undef):
User ID to log in to Sybase (default: sa):
Password (default: undef):
Note (probably harmless): No library found for
➥-lcs
Note (probably harmless): No library found for
➥-lsybtcl
Note (probably harmless): No library found for
➥-lcomn
Note (probably harmless): No library found for
➥-lintl
Using DBI 1.20 installed in
  /usr/lib/perl5/site_perl/5.6.0/i386-
➥linux/auto/DBI
Writing Makefile for DBD::Sybase
```

Ebben a részben a `Makefile.PL` adatokat kér be a Sybase-kiszolgálóról. A bekért adatokat egy ***PWD*** nevű fájlba írja ki, melyet a próbák futtatása során használ fel. Ezeket a teszteket Sybase-kiszolgálóhoz tervezték, nem Microsofthoz, így a próbák nálam sem voltak sikeresek. A könyvtárak hiányára utaló üzenetek azért jelennek meg, mert a Sybase könyvtárai helyett FreeTDS könyvtáraival fordítottuk le a rendszert. Ezután futtasd a `make`, majd a `make install` parancsokat.

A `make install` utolsó néhány üzenete valahogy így fog kinézni:

```
Installing /usr/share/man/man3/DBD::Sybase.3
Writing /usr/lib/perl5/site_perl/5.6.0/i386-
➥linux/auto/DBD/Sybase/.packlist
Appending installation info to
➥/usr/lib/perl5/5.6.0/i386-linux/perllocal.pod
```

A DBD::Sybase telepítése befejeződött. Most pedig állítsuk be a FreeTDS-t, hogy az SQL 2000-es adatbázisodhoz tudjon kapcsolódni. A FreeTDS beállítóállományát értelemszerűen ***freetds.conf***-nak nevezik. A fájlt a `freetds` könyvtárai alatt találod meg a ***/etc*** könyvtárban, melynek teljes elérési útja az én esetemben: ***/usr/local/freetds/etc/freetds.conf***. Ez a fájl már példát tartalmaz a Microsoft kiszolgálóbeállításaihoz, és csak annyiban kell módosítani rajta, hogy megfeleljen kiszolgálóid beállításainak. Az én beállítófájlom így néz ki:

```
# Egy jellegzetes Microsoft SQL-kiszolgÆl  7.0
# beÆll tÆs
[file1]
      host = file1
      port = 1433
      tds version = 7.0
```

A saját SQL-kiszolgálómat ***file1***-nek hívják és az 1433-as kapun keresztül érhető el, a TDS 7.0-s változatának használatával. A beállítóállomány elején egy általános beállításokat tartalmazó rész is található, a TDS változata ott is megadható. Győződj meg róla, hogy a kiszolgálód fel tudja oldani adatbázis-kiszolgálód hostnevét, vagy pedig egyszerűen csak IP-címet adj meg. Most már készen állunk egy kapcsolat létrehozására. *Listánk* egy olyan parancsfájlt tartalmaz, melyet ***Trevor Price*** munkatársam írt. Ez a parancsfájl egy Northwind nevű adatbázist kérdez le, melyet egy SQL 2000-telepítés alapértelmezés szerint tartalmaz. Kollégám tárolt eljáráshívást használt fel, az `sp_help`-et, mely a pillanatnyi adatbázisban található összes tábláról tájékoztatást ad. Másold ezt a szöveget egy tetszőleges fájlba, aminek a neve legyen mondjuk ***testsql.pl***, majd pedig a saját beállításaidnak megfelelően szerkeszd át a `$user` és a `$password` változókat. A parancsfájl futtatásával a következő kimenetet kapod:

```
perl testsql.pl
rows is -1
Alphabetical list of products dbo      view
Category Sales for 1997 dbo      view
Current Product List dbo      view
Customer and Suppliers by City  dbo       view
Invoices     dbo      view
Order Details Extended dbo      view
```

Gratulálok, hogy eljutottál idáig! Remélem, neked is olyan hasznos volt ez a rész, mint nekünk!

Linux Journal április, 96. szám

Andrew Trice
a Vital Link Business Systems rendszerfelügyelője, egyúttal az Iron Robot Records vezető mérnöke a cég független San Francisco-i kirendeltségében. Andrew a Cornell Egyetemen szerzett bölcsészdiplomát angol irodalomból.

Zope-termékek készítése

Reuven ebben a hónapban elmeséli, hogyan készíthetünk egyszerű Zope-termékeket, illetve miképpen illeszthetjük be őket honlapunkba.

Múlt hónapban folytattuk a Zope webfejlesztő környezetben tett kirándulásunkat, és megvizsgáltunk, illetve telepítettünk néhány Zope-terméket. Mint láthattuk, minden termék tulajdonképpen egy-egy Python-objektummodul, amelyet azután egy vagy több példányban létrehozhatunk a Zope-kiszolgálón. Termékek százait tölthetjük le a Zope-hoz, kezdve az apró, pehelysúlyú szerencsemondótól egészen a hatalmas és lenyűgöző tartalomkezelő keretrendszerekig (content management framework – CMF).

Számos Zope-pal dolgozó rendszergazda meg is elégszik a Webről letöltött anyagok telepítésével és használatával. Kétségtelen, hogy átlagosan egyszerű honlapigényeink kielégítéséhez bőven található termék a neten; amit pedig nem tudunk termékek segítségével megoldani, az többnyire elég egyszerű ahhoz, hogy megírhassuk DTML-ben, a Zope Dynamic Template Markup Language nyelvén (erről a témáról bővebben a Linuxvilág 2002 februári számában olvashatunk).

Bármennyire is egyszerű és magától értetődő bizonyos dolgokat DTML-ben megoldani, soha nem olyan teljes és rugalmas, mintha Python alatt készítenénk. Igaz ugyan, hogy a Python-Scripts (és a PerlScripts) megjelenése a Zope-ban számos közepes méretű feladatban szükségtelenné tette termékek készítését, mégis a legtöbb Zope-programozó előbb-utóbb azon kapja magát, hogy valamilyen terméket ír.

Ebben a hónapban megvizsgáljuk, hogyan írhatunk egyszerű Zope-terméket, amit aztán beépíthetünk a honlapunkba. Mint látni fogjuk, igen könnyű a környezet többi részéhez jól illeszkedő Zope-terméket készíteni.

Egy igen egyszerű termék

A Zope-termékek lényegében Python-modulok. A termékek – mint a múlt hónapban láthattuk – a fő Zope könyvtár ***lib/python/Products*** alkönyvtárába kerülnek. A Zope a termékeket csak induláskor és újraindításkor nézi meg, így minden új termék telepítése után a Zope-ot újra kell indítanunk.

A telepített termékeket ezután tetszés szerinti számban létrehozhatjuk, minden példányt elhelyezve valahol a honlapszerkezetben. Minden példány egyedi azonosítóval (ID) rendelkezik, amely egyrészt egyértelműen azonosítja őket a könyvtárban, másrészt lehetővé teszi, hogy az objektum eljárásaira hivatkozzunk.

Lehet, hogy kicsit zavarosan hangzik, de emlékezzünk arra, hogy a */foo/bar* URL általában azt jelenti, hogy a webkiszolgáló a *foo* könyvtárban található *bar* állományt adja vissza. Ezzel szemben Zope alatt a *foo/bar* URL azt jelenti, hogy a rendszernek a `foo` objektum `bar` eljárását kell meghívnia. Más szavakkal a *foo/bar* a `foo.bar` kifejezéssé alakul át.

Továbbá amikor azt mondjuk, „a foo objektum”, valójában azt kellene mondanunk, hogy „a foo azonosítójú objektum”. Az azonosító (ID) beállítása nélkülözhetetlen, ha az objektum eljárásait sikeresen meg szeretnénk hívni.

Termékünkhöz egy `meta_type` értéket is meg kell adnunk. A `meta_type` név lesz az a szöveg, amely a */**manage*** képernyő jobb felső sarkában fellelhető Zope-terméklista ***Add*** nevű lenyíló menüjében szerepelni fog. Általában ugyanazt a nevet adhatjuk a `meta_type`-nak is, amit az osztályhoz is használtunk, de valami könnyebben érthető azonosítót is használhatunk.

Ne feledjük, hogy az ***Add*** menü listája ASCII sorrend szerint rendezett, ami azt jelenti, hogy a nagy ***Z*** előbb következik, mint a kis ***a***.

Termékünk létrehozásához a következő lépéseket kell megtennünk:

- Megadunk egy külön modulban elhelyezkedő osztályt és a ***lib/python/Products*** könyvtár alá telepítjük.
- Megadunk egy `__init__` eljárást, amelyben az `id` példányváltozóhoz valamilyen értéket rendelünk.
- Megadunk egy vagy több eljárást, amelyeknek a visszatérési értéke HTML-t tartalmazó szöveg.
- Megadunk egy `meta_type` osztályváltozót, amely azután termékünk összes példányánál beállítja a `meta_type` értékét.

Láthatjuk, hogy mennyire egyszerűen hozhatunk létre termékeket – az *1. lista* jól szemlélteti a folyamatot. Itt megadjuk a ***helloworld.py***-t, ezt az egyszerű Zope-terméket, amelyet ezután már telepíthetünk, és hajszál híján példányokat is készíthetünk belőle (hamarosan azt is megtudjuk, hogyan lehetünk úrrá ezen a hiányosságon).

Akad még néhány lényeges elem, amit érdemes megvizsgálni `helloworld` osztályunkban. Elsőként mind az osztály, mind az eljárások leírást is tartalmaznak. Soha nem árt leírást készíteni, és az a tény, hogy a Pythonban egy ilyen beépített szolgáltatást találunk, ritka, de figyelemreméltó emlékeztető: a programozók jól teszik, ha hajlandók leírást fűzni a forráskódjukhoz. A Zope ezt az ajánlást kötelezővé teszi azáltal, hogy a rendszerben használt összes eljárás esetében megköveteli a dokumentációs sorok jelenlétét.

A `helloworld` osztályunk két eljárást is megad: az `__init__` és az `index_html` eljárást. Az `__init__` eljárást a Zope önműködően hívja meg, amikor az objektumunkból példányokat készít, és általában arra használjuk, hogy alapértéket adjunk példányváltozóinknak, illetve a később szükséges jellemzőket is itt adhatjuk meg. Jelen esetben az `__init__` egyetlen példányváltozónak ad értéket (`self.id`), amely lehetővé teszi, hogy objektumunk nyomon követhesse a saját „személyazonosságát”. Ahogy az várható is, az `__init__` eljárás nem arra való, hogy a külső világból hívjuk meg, ezt az eljárást a Zope-nak magának kell meghívnia.

Az `index_html` eljárás feladata ezzel szemben az, hogy a

1. lista A helloworld.py egy igen egyszerű Zope-termék

```
class helloworld:
  "Ez egy pØlda Zope-termØk,
   a ·helloworld· osztÆly"

  meta_type = "helloworld"

  def __init__(self):
     "Ez az eljÆrÆs h v dik meg, amikor œj
      ➥helloworld-pØldÆny j n lØtre"
     self.id = id

  def index_html(self):
     "AlapØrtelmezØs szerint ez az eljÆrÆs
      ➥h v dik meg a k nyvtÆrban"
     return """<html>
     <head>
     <title>Hello, world!</title>
     <body>
     <h1>Hello, world!</h1>
     <p>Ez az egyszerß Python-termØk nk
     ➥kimenete</p>
     </body>"""
```

címén keresztül hívjuk meg. Ha a ***helloworld*** egy példányát a Zope-kiszolgáló fő könyvtárába (/) helyezzük, az `index_html` eljárást a ***/helloworld/index_html*** cím segítségével hívhatjuk meg. Az `index_html` azonban különleges: a sok Apache kiszolgálón használt ***index.html*** állományhoz hasonlóan – ha nincsen más eljárás megnevezve – alapértelmezés szerint ez indul el.
Végül figyeljük meg, hogy az `index_html` HTML-t ad vissza hívójának. Nem ad vissza állapotkódot vagy bármi mást a HTML-en kívül.

Mi hiányzik?

A ***helloworld.py*** tökéletesen szabályos Zope-termék; fel is telepíthetjük a ***lib/python/Products*** könyvtárba és a Zope nem fog tiltakozni. Ugyanakkor a Zope sajnos mégsem veszi észre, hogy a ***helloworld.py*** itt helyezkedik el, nem helyezi az ***Add*** választéklistába sem, és általában figyelmen kívül hagyja az összes munkát, amit a termék megírásába fektettünk. Nyilvánvalóan egy kicsit fel kell hízlalnunk csontváztermékünket, ha a Zope-pal szeretnénk kapcsolatot tartani. Ezt a továbbfejlesztett változatot ***smallhello*** terméknek nevezzük el.
Első lépésként át kell alakítanunk az egyetlen modulfájlból álló (***helloworld.py***) termékünk szerkezetét teljes értékű Python-csomaggá. A csomag egy könyvtár (***smallhello***) a Python keresési ösvényén (search path), és a `sys.path` változó adja meg. A modulfájlban egy vagy több Python-forrásfájl található. A mi esetünkben a ***smallhello*** könyvtár két fájlt fog tartalmazni: a ***smallhello.py*** állományt, amely meglehetősen hasonlít a ***helloworld.py***-re (lásd a *2. listát*) és az ***__init__.py***-t (lásd a *3. listát*), amely alaphelyzetbe állítja és segít bejegyezni az objektumunkat.
A ***__init__.py*** fájl először beolvassa a `smallhello.smallhello`-t, amely meghatározza a modul eljárásait és attribútumait. Az ***__init__.py*** leglényegesebb része azonban, legalábbis a Zope szemszögéből, a beállító (initialize) eljárás. Miután a Zope megtalálta és beolvasta a `smallhell t`, meghívja a `smallhello.initialize`-t – értékként átadva neki a `ProductContext` objektumot (amit „context”-nek nevez). Más szavakkal az objektum alaphelyzetbe állítása azt eredményezi, hogy az objektum bejegyzi magát a kiszolgálón.
Az alaphelyzetbe állító függvény meglehetősen egyszerű, pedig a mi változatunk még alapvető hibakezelést is végez (a try–except páros használatával) a dolgok helyes működésének megteremtéséért. A smallhello termék mindössze két értéket ad át a `context.registerClass` osztálynak: a `finalhello.finalhello` objektumot, amelyet fel szeretnénk venni, illetve egy `constructor` csoportot, amelyet akkor kell meghívni, amikor új termékpéldányt hozunk létre. Ne feledjük el kitenni a befejező pontot, ha csupán egyetlen elemet adunk át constructorként; ha nem így teszünk, a Zope nem fogja tudni betölteni a terméket.
A `constructors` érték csak az egyik a számos érték közül, amit átadhatunk a `context.registerClass` osztálynak, s amelyekkel testreszabhatjuk objektumunk Zope-bejegyzését. Például átadhatunk egy ikon (`icon`) értéket, ezáltal tudatva a Zope-pal, hogy melyik képet (azaz a csomag könyvtárában található fájlnevet) szeretnénk csomagunk példányai mellett látni a Zope-ban.

Az Objektum módosítása

A ***helloworld.py smallhello.py***-vé változtatása (2. lista) néhány apró módosítást is igényel. Kezdjük egy új eljárás létrehozásával, amely lehetővé teszi, hogy a Zope-termékünkből új példányokat hozzon létre. Hagyományosan az ilyen kezeléshez kapcsolódó (management-related) eljárások a `manage_` előtaggal kezdődnek, így eljárásunkat `manage_smallhello`-nak fogjuk elnevezni. Ez ugyanaz az eljárás, mint amit a `context.registerClass`-nak átadott `constructors` csoportban is megneveztünk.
A `smallhello` osztályunkon végzett legjelentősebb változtatás egyben az egyik legkevésbé nyilvánvaló: egy alosztályt készítettünk a Zope-csomag (OFS.SimpleItem csomag) részeként elérhető Zope termék-alaposztályok `OFS.SimpleItem.SimpleItem` osztályából. A `SimpleItem` alosztályaként örökölhető tulajdonságok nélkül számos dolog – például az objektumok „fogd és vidd” alapú áthelyezése – egyáltalán nem úgy fog működni, ahogy elvárnánk. Létezik néhány alaposztály, amelyektől termékünk egysmást örökölhet; a `SimpleItem`, mint a neve is mutatja, társai közt a legegyszerűbb és legkönnyebben érthető.
Miközben megváltoztattam a ***smallhello.py***-t, úgy döntöttem, hogy további két tartalomkészítő eljárással is megtoldom. Az egyikük az `other_html`, az `index_html`-hez hasonló tartalmat hoz létre – kivéve természetesen, hogy ha nincsen más eljárás megadva, az `index_html` fog megjelenni; míg az `other_html` csak akkor látszik, ha az URL-ben közvetlen módon megnevezzük.
Ezenkívül beillesztettem egy `foo_file` eljárást is, amely azt mutatja be, hogyan adjunk vissza HTML-tartalmat (vagy DTML-t) a merevlemezről. Egy kicsit bosszantó és kiábrándító lehet, ha minden HTML-tartalmat Python-modulfájlokba kell tennünk; így a DTML-fájlokat egyszerűen csak a csomag könyvtárába kell helyeznünk, a módosításokat viszont már a programtól teljesen függetlenül végezhetjük. Figyeljük meg, hogy ehhez a Globals csomagból a `HTMLFile` eljárást be kellett importálnunk.
A ***smallhello.py*** `__init__` függvényét úgy módosítottam,

2. lista A smallhello/smallhello.py

```
import OFS.SimpleItem          # Alap lekØrØse
from Globals import HTMLFile   # ´gy hozhatunk
                               # majd be
                               # HTML-fÆjlokat

class smallhello(OFS.SimpleItem.SimpleItem):
    """ ez az osztÆly hatÆrozza meg a
    smallhello termØket. LØtezik egy
    alaphelyzetbe Æll t  eljÆrÆsunk
    (__init__), egy mÆsik, amely alapesetben
    egy r vid HTML- zenetet jelen t meg, Øs egy
    harmadik, amely egy DTML-fÆjl tartalmÆt
    jelen ti meg."""

    meta_type = ·smallhello·

    def __init__(self, id, title):
        "Smallhello œj pØldÆnyÆnak
        alaphelyzetbe Æll tÆsa"
        self.id = id
        self.title = title

    def index_html(self):
        "LÆssunk nØmi alapvetı tartalmat!"
        return """
        <html>
        <head><title>Hello, world!</title></head>
        <body>
        <h1>Hello, world!</h1>
        <p>Welcome to my ·smallhello· Zope
        ➥product</p>
        </body>
        </html>"""

    def other_html(self):
        "mØg egy kis egyszerß tartalom"
        return """
        <html>
        <head><title>More content!</title></head>
        <body>
        <h1>More content!</h1>
        <p>You can define lots of methods if
           you want...</p>
        </body>
        </html>"""

    def foo_file(self):
        "Bemutatjuk, hogyan tudunk tartalmat
         megjelen teni fÆjlb l"
        return HTMLFile(·foo·, globals())

def manage_smallhello(self, RESPONSE):
    "Smallhello hozzÆadÆsa a k nyvtÆrhoz."
    self._setObject(·smallhello_id·,
    smallhello(·smallhello_id·,
    ·smallhello_title·))
    RESPONSE.redirect(·index_html·)
```

hogy három értéket fogadjon. Ezek: a `self`, az `id` és a `title` (korábban csak a `self` és `id` értékeket használtuk). Az `__init__` függvény minden új ***smallhello.py*** példány létrehozásakor meghívódik, amit egyébként a `manage_smallhello` hívással érhetünk el.
A `manage_smallhello` belsejében a `self._setObject` hívásunk az objektumazonosítót az általános `smallhello_id`-re állítja, kiegészítve a `smallhello_title` címmel. Mivel példánkban az azonosítót beleégettük a kódba, és mivel az azonosítóknak minden könyvtárban egyedieknek kell lenniük, ez azt jelenti, hogy smallhello termékből egy adott könyvtárba csak egyetlen példányt helyezhetünk. Sajnos kevés a hely az értékírás és olvasás ismertetésére, azonban a ***Kapcsolódó címek*** között megemlített példák gyors átfutása könnyen érthetővé teszi, hogyan kell az ilyesmit elkészíteni.
A `self._setObject` meghívása után a `manage_smallhello` a felhasználó böngészőjét átirányítja a fő (`index_html`) eljárásra. Itt valamilyen tájékoztatást is megjeleníthetnénk a felhasználó böngészőjét objektumunk egy másik eljárására irányítva, én azonban inkább az egyszerűbb utat választottam, és a felhasználókat a lapunk ***/index.html*** oldalára küldöm.
Miután a smallhello terméket telepítettük, újra kell indítanunk a Zope-ot. Most már látnunk kell a `smallhello` elemet az ***Add*** menü alja környékén; kiválasztva Zope-honlapunk `index.html` oldalán találjuk magunkat. Mivel termékünket nem tettük túl felhasználóbaráttá, az URL-t (`index_html`, `other_html` vagy `foo_file`) kézzel kell a böngészőbe begépelnünk. Természetesen semmi akadálya nincsen, hogy ezek a lapok néhány egymásra való hivatkozást tartalmazzanak, illetve a honlaprendszer más lapjai ne ide mutassanak.
Mit szólsz? Készítettünk egy Zope-terméket!

Mi hiányzik még mindig?

Ha jelenlegi formájában akarnánk kiadni simplehello projektünket, nemigen akarná senki sem használni. A fent említett nehézségeken kívül (például az egyes példányoknál az egyedi azonosítók hiánya) termékünk nem tartalmaz kezelőtáblákat sem (`management tabs`), amelyek a Zope-ot a rendszergazdák számára oly felhasználóbaráttá teszik.
A biztonsági jogosultságokat sem kezeli igazán szabványos és egyszerű módon.
Ezeket a képességeket is majdnem olyan könnyű telepíteni, mint amelyeket eddig láttunk. Például minden tábla egy-egy könyvtárat jelent, amelyben két név-érték pár helyezkedik el, a címke (label) és a művelet (action). A címkéhez rendelt értéket látja a felhasználó a képernyőn, míg a művelethez rendelt érték határozza meg, hogy a Zope melyik eljárást fogja meghívni, ha valaki az adott táblára kattint. A táblák telepítéséhez adjunk meg egy `manage_options` csoportot (tuple) objektumunkban, amelynek elemei a táblát írják le.
Az egyik leglényegesebb elem, amiről eleddig még nem beszéltünk: a felhasználói bemenet. Valójában ezt könnyű megoldani, mivel a Zope a HTML-űrlapokat úgy kezeli, mintha az eljárás normál értékei lennének. Például vegyük a következő HTML-űrlapot:

3. lista A smallhello/__init__.py

```
# Az osztÆlyfÆjl importÆlÆsa
import smallhello

# eljÆrÆs, amely œj smallhello-pØldÆnyt hoz
# lØtre
def initialize(context):
  "TermØk nkbıl œj pØldÆnyt hozunk lØtre"

  # OsztÆlyunk (product) bejegyzØse a
  # jelenlegi
  # acquisition contextben, ahol megmutatjuk
  # milyen eljÆrÆs (vagy eljÆrÆsok)
  # h v djanak meg, amikor valaki œj
  # pØldÆnyt kØsz t a termØk nkbıl.

  # A "Boring" pØldamodulban fellelhetı
  # tr kk t alkalmazzuk, amely kivØteleket
  # hasznÆl a termØk bejegyzØse sorÆn
  # felmer lı hibÆk elfogÆsÆra.

  try:
    context.registerClass(

      # Milyen objektumot adunk hozzÆ?
      smallhello.smallhello,

      # Milyen eljÆrÆst kell megh vni,
      # amikor œj smallhello-pØldÆnyt
      # szeretnØnk kØsz teni?
      constructors =
      ➥(smallhello.manage_smallhello,)
       )

  except:
    # Ha valami gond van, jelents k a
    # stderr csatornÆn (ahogy ez a Boring
    # bemutat termØkben is t rtØnt)


    # ImportÆljuk a teljes k rß hibakeresı
    # adatokat elØrhetıvØ tevı modulokat
    import sys, traceback, string

    # MegÆllap tjuk, mi volt a gond.
    type, val, tb = sys.exc_info()

    # Ismertetj k a felhasznÆl val az okot
    sys.stderr.write(string.join(
      traceback.format_exception(type, val,
      ➥tb), ··))

    del type, val, tb
```

```
<form action="manage_edit" method="POST">
    <p>id: <input type="text" name="id"></p>
    <p>Title: <input type="text"
    ➥name="title"></p>
    <p><input type="submit"></p>
</form>
```

Miután a ***Submit*** gombra kattintottunk, termékünk `manage_edit` eljárásának két név-érték párost adunk át `id` és `title` néven. Az eljárást a következőképpen adhatnánk meg:

```
def manage_edit(self, id, title):
```

Ebben az eljárásban az azonos nevű változók felhasználásával lekérhetjük az `id` és a `title` HTML-űrlapelemek értékét.

Összegzés

A Zope-termékek a DTML-fájloknál sokkal kifinomultabb és fejlettebb lehetőséget kínálnak a Zope-alkalmazások létrehozására. Nagyobb rugalmasságot tesz elérhetővé, ám egyben nagyobb fegyelmet és a rendszer magasabb szintű ismeretét is igényli. A Zope-termékek írásának ismerete olyasmi, mintha `mod_perl`-modulokat írnánk az Apache-hoz; ez azt jelenti, hogy az alaprendszer teljes egészében a rendelkezésünkre áll. A programozókat igen gazdag API segíti saját Zope-termékük elkészítésében, a jó leírás hiánya azonban sajnos sokakat elriaszt a próbálkozástól. Saját simplehello termékünk bemutatta, hogy igen rövid idő alatt pár sornyi kóddal is jelentős és hasznos alkalmazásokat tudunk létrehozni.

Linux Journal április, 96. szám

Kapcsolódó címek

A nyílt forrású Zope-terméket a Zope Corporation (Digital Creations) készíti és tartja karban. Bővebb információ a Zope honlapján található ➲ http://www.zope.com

A Zope egyik tagja MaxM által készített egyszerű kétrészes „kis termék” készítői leírás elérhető a ➲ http://www.zope.com/Members/maxm/HowTo/minimal_01/source címen.

A Pythonról többet a ➲ http://www.python.org oldalon olvashatunk. Két jó Python-könyv a Learning Python (*Mark Lutz* és *David Ascher* az O'Reilly kiadásában) és a Python Essential Reference (*David M. Beazley* és *Guido Van Rossum* a New Riders kiadásában).

Reuven M. Lerner (reuven@lerner.co.il) kisebb, webes és internetes módszerekkel foglalkozó tanácsadó cég tulajdonosa és vezetője. A cikk megjelenésének időpontjában valószínűleg már végleg elkészült Core Perl című könyvével, melyet idén jelentet meg a Prentice-Hall. Az ATF honlapon érhető el (➲ http://www.lerner.co.il/atf/).

Egyre közelebb a rendszermaghoz

Marcel a Linux-rendszermag megfigyelésének különböző lehetőségeit mutatja be.

François, mon ami, még mindig a Szabad Linux Rádió rendszermagról szóló Vorbis-adását hallgatod? Bár csodálom elhatározásodat, hogy e havi beszélgetésünkre megismerkedsz a Linux-rendszermag működésének mélységeivel, félek, hogy a teljes adás hónapokig elhúzódhat. Érdekesebb úton is közel juthatsz a rendszermaghoz, François, anélkül, hogy elveszettnek tűnnél. Mindemellett vendégeink is mindjárt megérkeznek, addigra készen kell állnunk.

Á, bonjour, mes amis! Jó látni titeket. Isten hozott Chez Marcelnél, a finom Linux-fogások, kitűnő borok, jó ételek és a nagyszerű társaság házában. Ez a társaság természetesen ti vagytok. Helyezzétek magatokat kényelembe. Hűséges pincérem, François mindjárt leszalad a pincébe borért. Akad egy üveg 1999-es Cornas Champelrose-unk, mely remekül illik a mai menühöz. Ha már a menü szóba került, azt szeretném, ha úgy tekintenénk rá, mint egy önkiszolgáló étteremre vagy falatkákkal megrakott svédasztalra, válogatva a kicsi, de hasznos programok között, melyekkel munkafelületünket fűszerezhetjük. Á François, visszaérkeztél, remek. Kérlek, tölts a vendégeinknek!

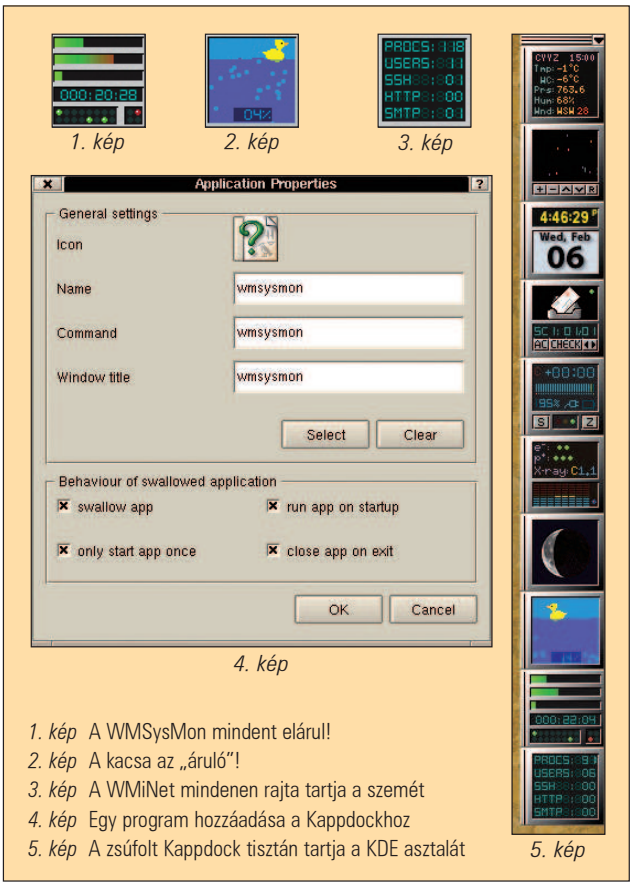
Mielőtt megérkeztetek, mes amis, éppen azt fejtettem ki Françoisnak, hogy talán akkor kerülhetünk legközelebb a Linux-rendszermag lényegéhez, ha kívülről vizsgáljuk. Végül is a rendszermag a Linux-rendszer szíve-lelke, ez biztosítja az összes többi program működését. Ha azt szeretnénk tudni, hogy mit tesz az adott pillanatban, a rendszermegfigyelő eszközöket használhatjuk. Ilyen programból, ahogy már bizonyára tapasztaltuk is, több száz létezik, ezért nem könnyű választani közülük. Én Linux-konyhámban időről időre váltogatom a futtatott munkaasztalt, de legtöbbször a Window Makert és a KDE-t használom. Főleg a KDE hatékonysága és szépsége ragadott meg, de a Window Maker is nagyon kedves a szívemnek, ami leginkább a dock-alkalmazások könnyedségének köszönhető. Ezek a kis 64×64 képpontos programok minden területet lefednek, még a rendszer- és erőforrás-felügyelő alkalmazásokét is. Vizsgáljuk meg közelebbről például ***Vito Caputo*** WMSysMon programját (amit eredetileg ***Dave Clark*** írt).

A program honlapja a ➲ http://www.gnugeneration.com/software/wmsysmon címen érhető el. Ez a programocska grafikusan jeleníti meg az éppen felhasznált memória mennyiségét, a csereterület és I/O százalékban kifejezett kihasználtságát, csakúgy, mint a megszakításokat és a csereterületre be- és onnan kiforgatott oldalakat. Rengeteg mindent megtalálhatunk ebben a kis felügyelőprogramban. Az *1. kép*, mely működés közben mutatja, képet ad arról, mit is várhatunk tőle.

A program fordítása nem áll többől, mint a forráskód kicsomagolásából, a program könyvtárába (***wmsysmon-0.7.6/src***) való mozgatásából, ezek után pedig a `make` és `make install` futtatásából.

A program indításához egyszerűen a `wmsysmon` parancsot kell beírnunk. A megszakítások kijelzéséhez választhatunk a mérőóra és a LED-es kijelző között. Mes amis, el kell mondanom, hogy engem a számítógépekben sok évvel ezelőtt éppen ezek a nagyon fontosnak tűnő, felvillanó lámpácskákkal teli kijelzők ragadtak meg, ezért a WMSysMon-t az `-l` kapcsolóval futtatom.

A WMSysMon profi kinézete ellenére is el kell azonban ismernünk, hogy Linux-konyhánkban szorgoskodva szívünk alkalmanként a könnyebb ételek felé húz. Ha a WMSysMon túl komoly felügyelőprogramnak tűnik, érdemes megfontolnunk a Bubblemon dock-alkalmazás használatát, amit a timecop honlapjáról a ➲ http://www.ne.jp/asahi/linux/timecop címről tölthetünk le.



1. kép A WMSysMon mindent elárul!
2. kép A kacsa az „áruló”!
3. kép A WMiNet mindenen rajta tartja a szemét
4. kép Egy program hozzáadása a Kappdockhoz
5. kép A zsúfolt Kappdock tisztán tartja a KDE asztalát

Úgy teszek, mintha biztos lennék abban, hogy a programot hasznosnak fogjátok találni, és mindjárt fordításának módjával kezdem:

```
tar -xzvf bubblemon-dockapp-1.4.tar.gz
cd bubblemon-dockapp-1.4
make
make install
```

A program futtatásához a `bubblemon` parancsot kell beírnunk. Sokan dolgoztak különböző bugyborékoló erőforrásfigyelő programok megalkotásán, a legzseniálisabb mégis az itt található aranyos sárga kiskacsa, amely egy bugyborékoló tavacskán úszkál. Minél jobban dolgoztatjuk a rendszert, annál több buborék éri el a felszínt, és amikor elkezd fogyni a rendelkezésre álló memória, a vízszint emelkedni kezd, míg a kiskacsa eltűnik a szemünk elől.

Mozgassuk az egérkurzort a BubbleMon felett, ennek hatására a kacsa és a tó elhalványul, és egy processzorhasználatot ábrázoló grafikon tűnik fel kiegészítve az utolsó 5, 10 és 15 perc átlagos futtatási statisztikáival. Még egy tipp: miközben a kacsa halványul, a jobb egérgombbal a program képén kattintva, elkaphatjuk az átmeneti képet, vagyis a kacsa szellemét láthatjuk úszni a processzorhasználatot ábrázoló grafikonon. Letilthatjuk a kacsa megjelenítését a `-d` kapcsolóval, ámde miért akarnánk ilyet tenni?

Természetesen rendszermagunkat sok egyéb dolog is lefoglalja: a felhasználókkal kapcsolatos teendők, a különféle folyamatok futtatása, az elektronikus levelek, a webkiszolgálóhoz érkező kérések és az egyéb hálózati kapcsolatok bonyolítása.

A WMiNET egy Window Maker alatt futó, ***Dave Clark***, ***Antoine Nulle*** és ***Martijn Pieterse*** által írt program, melynek célja (többek között) a különböző hálózati kapcsolatok, folyamatok és felhasználók felügyelete. A ➲ http://www.neotokyo.org/illusion címről tölthető le.

Miután a forrást a `tar -xzvf wminet-2.0.3.tar.gz` paranccsal kicsomagoltuk, váltsunk a ***wminet.app/wminet*** könyvtárra és adjuk ki a szokásos `make` és `make install` parancsokat. A telepítőfolyamat létre fog hozni a ***/etc/wminetrc*** könyvtárban egy beállítófájlt, amelyben megadhatjuk, hogy milyen rendszerfolyamatokra terjedjen ki a program figyelme. Különösen érdekesnek találtam, hogy a kijelző mind az öt sorához hozzárendelhető egy egykattintásos parancs. Az első sorban például a futó folyamatok listáját látom, ha erre a sorra kattintok, az önműködően elindítja a `top` parancsot. Alább a ***/etc/wminetrc*** fájlomból vett minta látható, azt hiszem, magától értetődő lesz a számotokra:

```
action1=rxvt -bg black-fg white -e top
action2=rxvt -bg black-fg white -e sh -c "w; read"
action3=rxvt -bg black-fg white -e sh -c"netstat
➥-etpn; read"
action4=rxvt -bg black-fg white -e tail-f
➥/usr/local/apache/var/logs/access_log
action5=rxvt -bg black-fg white -e sh -c
➥"df -k;read"
```

Régebben, amikor ezeket a Window Maker alatt futtatható kis dock-alkalmazásokat vizsgáltam, említettem, hogy más asztalokon is futtathatók. A KDE vagy hasonló rendszer alatt futtatva sajnos megvan az a hátrányuk, hogy a folyamatok a tálcán jelennek meg és a programok nem rendelkeznek saját kerettel. Jól működnek, azonban nélkülözik a Window Maker által biztosított eleganciát, amellyel a programokat egy helyre gyűjtve azok a rendszer lelkébe néző ablakok áttekinthető gyűjteményét adják.

Henning Burchart-nak és a Kappdocknak köszönhetően ezentúl a KDE felhasználóinak sem kell ezt a mellőzöttséget elszenvedniük. Ez a programocska csendben megül a sarokban, és várja, hogy ezeket a Window Maker alá írt programokat hozzákapcsoljuk. A Kappdock egy kis ikont is elhelyez a KDE ikontálcáján. A futó alkalmazásokat egyetlen kattintással eltüntethetjük a szemünk elől – egy újabb kattintás, és máris újra megjelennek. Be fogom mutatni a kezelését is, most azonban kezdjük Henning weboldalának felkeresésével a ➲ http://www.informatik.uni-oldenburg.de/~bigboss/kappdock címen. Bontsuk ki és fordítsuk le a forrást az ismerős kicsomagolás-beállítás-make feladathármas elvégzésével:

```
tar -xzvf kappdock-0.44.tar.gz
cd kappdock-0.44
./configure
make
make install
```

Indítsuk el a Kappdockot a háttérben a héjból futtatva, vagy az ALT-F2 billentyűpáros megnyomásával. A képernyőn egy ártalmatlan kis négyzetet kell észlelnünk csíkozott sávval a tetején, rajta egy jobbra mutató fekete nyíllal (egy új ikon is megjelenik a KDE ikontálcáján). A sávot valójában húzósávnak (drag bar) nevezik, mert ezzel húzhatjuk a programokat a kívánt helyre. Én inkább magát a dockot értem alatta. A fekete nyílra kattintva a Kappdock eltűnik a képernyőről a tálca ikonjára kattintva pedig újra megjelenik. Hogyan adhatunk hozzá programokat? Kezdjük egy jobb kattintással a dockhoz rögzített négyzeten, ekkor vagy a létező négyzetet szerkesztjük, vagy egy újat hozhatunk létre. Ha a jobb egérgombbal a húzósávon kattintunk, lehetőségünk nyílik új program hozzáadására vagy a Kappdock beállításainak megváltoztatására. Ezek közül néhány a dock helyzetét és a bennefoglalt programok tájolását határozza meg. Egy program hozzáadásához a ***New*** (új) menüpontra kell kattintanunk. A *4. képen* láthatóhoz hasonló párbeszédablakot fogunk kapni.

Ha már van egy futó programunk (a korábban említettek közül), egyszerűen rákattinthatunk a ***Select*** (kiválasztás) menüpontra, ennek hatására a kurzor célkereszt formát vesz fel, amit a futó alkalmazás fölé mozgathatunk és rákattinthatunk. Voilà! A programot már fel is vettük a dockba. A másik lehetőség, hogy a szükséges adatokat saját magunk gépeljük be.

Az *5. képen* egy jól megrakott Kappdockot láthatunk.

Mindez már a tiétek, mes amis.

Viszontlátásra a következő hónapban. A votre santé!

Bon appétit!

Linux Journal május, 97. szám

Marcel Gagné (mggagne@salmar.com) Mississaugaban (Ontario, Kanada) él, a Salmar Consulting Inc. cég elnöke. A cég rendszerépítéssel és hálózati tanácsadással foglalkozik. Marcel pilóta és író egy személyben (tudományos-fantasztikus regényeket ír), társszerzője egy scifi-, fantázia- és horrorantológiának, a TransVersionsnak. Kedveli a Linuxot és a Unix minden változatát. Mostanában a Linux System Administration: A User's Guide című művén dolgozik. A világhálón elérhető honlapján sok hasznos dolgot találhatunk. ➲ http://www.salmar.com/marcel/

Kapcsolódó címek

Ben Sinclair Dock App Warehouse-a
➲ http://www.bensinclair.com/dockapp
BubbleMon-dockapp a timecoptól
➲ http://www.ne.jp/asahi/linux/timecop
Kappdock ➲ http://www.informatik.uni-oldenburg.de/~bigboss/kappdock
A Linux Kernel Being Broadcast adása
➲ http://radioqualia.va.com.au/freeradiolinux
Marcel borlapja
➲ http://www.marcelgagne.com/nfwine.html
A WMiNet honlapja ➲ http://www.neotokyo.org/illusion
A WMSysMon
➲ http://www.gnugeneration.com/software/wmsysmon

Nyílt forrású 3D grafikus motor

Howard bemutatja a kereskedelmi 3D grafikus motorok nyílt forrású választási lehetőségét, a Crystal Space-t.

Szeretnél 3D-s játékot vagy alkalmazást készíteni? Legelőször 3D grafikai motorra lesz szükséged, amelyel mindezt felépítheted. Hagyományosan két választásod van: vagy kódolsz magadnak egy teljesen új motort, vagy magas használati díj fejében valamelyik ismert cég motorját használod. Ez utóbbi megoldás behatárolja végleges terméked kereskedelmi lehetőségeit.

Létezik azonban egy harmadik lehetőség is: a Crystal Space, egy 3D grafikus motor, amelyet ***Jorrit Tyberghein***, 31 éves belga programozó írt.

A Crystal Space-t Tyberghein 1997-ben készítette el, miután megismerte az olyan játékokat, mint a Doom és a Quake, és kíváncsi volt rá, hogyan is állítják elő őket. Nem lévén tapasztalata a grafikus motorok kódolásában, az Interneten keresett adatokat a 3D-s programozással kapcsolatban, majd mindössze két hónap alatt el is készítette a Crystal Space első változatát. Tyberghein a kódot nyílt forrásban elérhetővé tette elérhetővé, így kisvártatva megszületett a Crystal Space fejlesztői közösség. A grafikus motort az eredeti Linux-rendszerről azóta már Unix, 32-bites Windows, Windows NT és egyéb operációs rendszerek alá is átültették.

Jelenleg a Crystal Space-szel írt alkalmazások többsége szerepjáték. Mivel a motor egy vagy több kameraállás megvalósítását teszi lehetővé, valamint beépített hálózati támogatást tartalmaz, és nagymértékben megkönnyíti a többfelhasználós játékok – például a szerepjátékok – fejlesztését. Felhasználják még repülőszimulátorok, valós idejű stratégiai játékok és a Doomhoz vagy a Quake-hez hasonló saját nézetű lövöldözős játékok fejlesztése során is.

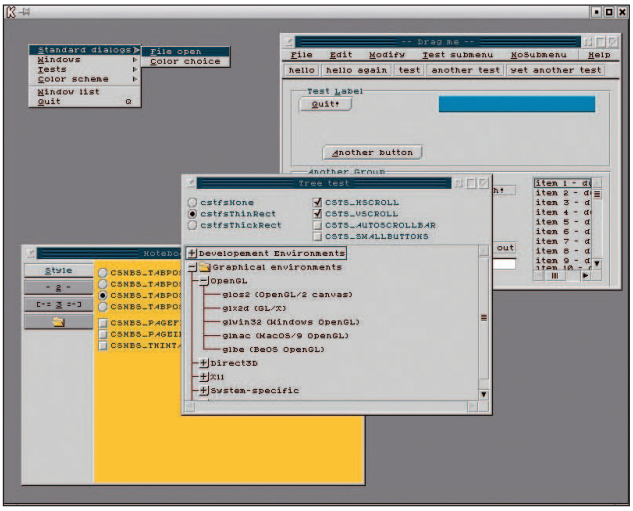
Ha azt vizsgáljuk, hogyan állja meg a Crystal Space a helyét a kereskedelmi 3D grafikus motorokkal szemben, főként a Quake III- és az Unreal Tournament-motorjával szemben, azt mondhatjuk, hogy bizonyos grafikai jellemzők terén összemérhető velük, de a kód kiforrottsága és sebessége tekintetében alattuk marad. Általánosságban a Crystal Space sokoldalúbb, mivel nem csupán játékok készítésére alkalmas. Gyakorlatilag egy olyan általános célú grafikus API, amit egy multimédiás böngésző, egy hangszerkesztő és egy képnézegető program készítéséhez is felhasználtak.

„A motornak számos olyan jellemzője van, amire a Quakenek vagy az Unrealnek sosem lesz szüksége, mivel az ilyen típusú játékok a grafikai szolgáltatásoknak csupán szűk körét használják” – állítja ***Andrew Zabolotny***, 28 éves szentpétervári programozó, aki a Crystal Space számára készít magas szintű kódokat.

A Crystal Space jellemzői: a Jó és a Rossz

A Crystal Space fejlesztői a motor megbízhatóságát (szinte sosem omlik össze) és a 3D-s leképezés (rendering) teljes megvalósítását emlegetik a fejlesztésben való részvétel és a használat melletti okokként.

Mennyire van létjogosultsága jelenleg egy 3D grafikus motornak, amikor a fejlett és erőteljes 3D grafikus alkatrészek már szinte alapvető tartozékai a legtöbb PC-nek? Az egyszerű válasz az, hogy még a legjobb 3D-s technológia mellett is szükségünk van egy programból megvalósított grafikus motorra, amely bizonyos mértékben megszabja az alkatrész működését.



1. kép Egy próbaalkalmazás, amely a Crystal Space ablakozórendszer számos jellemzőjét mutatja

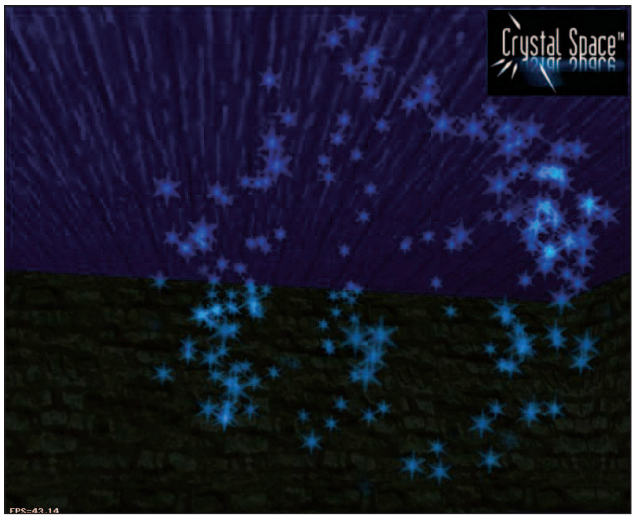
„Mindig megvan a határa annak, hogy az alkatrész mire képes” – mondja Tyberghein, a Crystal Space atyja. „Még a legfejlettebb alkatrész is nehezen boldogulna, ha sokszögek millióit kellene leképeznie. A grafikus motor feladata az hatékonyabbá tétel, vagyis behatárolja, hogy a grafikus kártyához egyáltalán mi jut el.”

Például programból megvalósított grafikus motorra van szükség annak meghatározására, hogy a játék virtuális környezetének mely részei láthatók egy adott pillanatban a felhasználó szemszögéből. Így a 3D-s alkatrész teljesítménye hatékonyabb lesz, hiszen nem kell erőforrásokat pazarolnia arra, hogy olyan objektumot képezzen le, amely ténylegesen nem is jelenik meg a képernyőn, még ha elméletileg létezik is (csak a játékos szemszögéből nem látható) a virtuális környezeten belül.

A Crystal Space az LGPL alatt ingyenesen terjeszthető. Ez a C++ nyelven írt 3D-s fejlesztőkörnyezet számos grafikus szolgáltatást és látványos jellemzőket támogat:

- tényleges hat szabadságfok;
- színes megvilágítás;
- mipmapping;
- kapuk;
- tükrök;
- alfaátlátszóság;

2. kép A Crystal Space által leképezett belső környezet



3. kép Csillagok, melyeket a Crystal Space részecskeanimáló rendszere képezett le

- tükröződfelületek;
- keretalapú és vázanimált 3D-s szerkezetek;
- eljárásalapú textúrák;
- radiosity;
- részecskerendszerek;
- volumetriás köd;
- parancsállományok készítése Python vagy egyéb nyelveken;
- 8-, 16- és 32-bites megjelenítés támogatása;
- Direct3D,
- OpenGL és programból megvalósított grafikaleképezés;
- betűkészlet-támogatás;
- hierarchikus transzformációk,

hogy csak néhányat emeljünk ki.

Bár a teljesítmény terén nem veheti fel a versenyt a Quake III vagy az Unreal Tournament motorjával, a Crystal Space-nek mégis megvannak a maga előnyei: több rendszeren is használható, vagyis olyan kódot készíthetünk, amely ugyanolyan jól fut Linux, Unix, 32-bites Windows, Windows NT és hét egyéb operációs rendszeren, amelyre a grafikus motort átültették.

A Crystal Space rugalmasan bővíthető, azaz egyetlen futtatható állomány is olyan különböző leképezőkkel képes működni, mint amilyen az OpenGL, Direct3D és a Glide. A grafikus motor első kiadása óta az OpenGL-leképezőt újraírták, így előző változatainál gyorsabban fut. Lényeges jellemző, hogy a Crystal Space sok 3D grafikus állomány-formátumot képes magától felismerni és olvasni. Lehetőség nyílik egyéb 3D-formátumok importálására is (például 3DS, OBJ, MDL, MD2, LWO és ASE). Megfordítva: a motor olyan

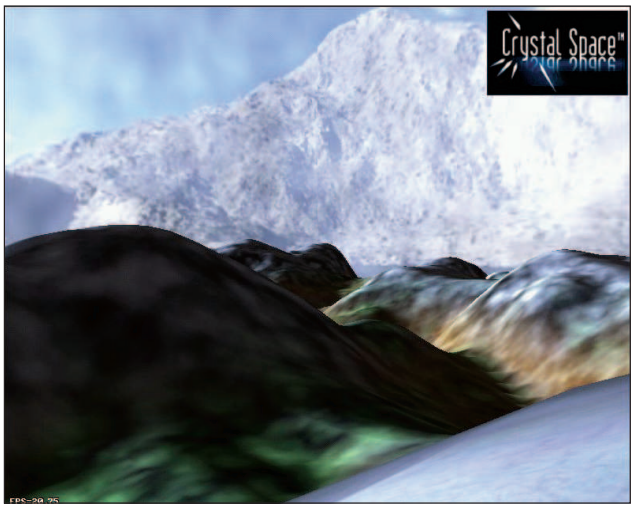
4. kép Egy Föld felé tartó űrhajó, amelyet a Crystal Space valós időben képezett le

Python-programokkal rendelkezik, amelyekkel a környezet és a modellek a Blender programból exportálhatók a Crystal Space-be. Bár a motor elsődleges célja 3D-s grafikák létrehozása, emellett még egy 2D-s API-val is rendelkezik. „Megírtam egy teljes körű ablakozó GUI-t, ami a Crystal Space alacsony szintű API-ján alapul” – mondta Zabolotny.

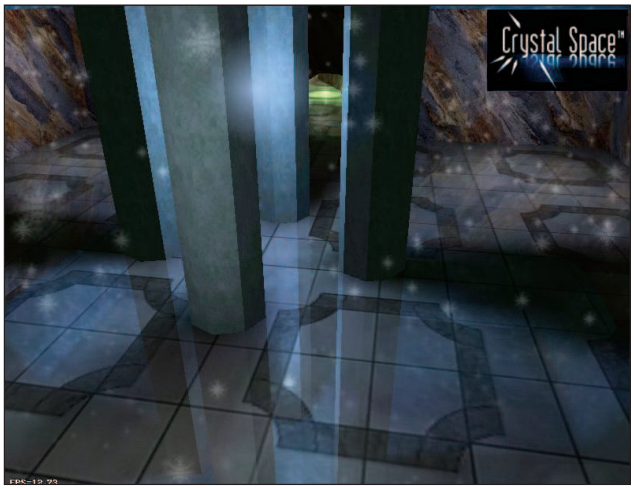
Különálló kódrészleteket használhatunk a Crystal Space-környezeten kívül is olyan munkafolyamatokban, amelyek nem játék- vagy multimédiafejlesztésekkel kapcsolatosak. Ezek között található egy csIniFile-osztály (***.ini*** állományok kezeléséhez), egy SCF (osztottosztály lehetőség) alrendszer, egy csArchive-osztály (***.zip*** állományok kezeléséhez) és egy VFS (virtuális állománymrendszer) alrendszer.

A Crystal Space legnagyobb gyengesége, hogy hiányzik belőle az ütközések felismerését megvalósító programrész. ***Thomas Hieber***, aki a Crystal Space segítségével egy Crystal Shooter nevű, saját szemszögű lövöldözős játékot fejleszt, ideje nagy részét azzal tölti, hogy a motor ezen képességét igyekszik fejleszteni. „Bizonyos mértékű támogatás ugyan létezik a motorban, de ez nem túl hasznos a játékok számára” – mondja a 30 éves német programfejlesztő. „Csak a statikus objektumok ütközésének ellenőrzése támogatott, ami arról nyújt információt, hogy hol történt az ütközés – ha történt egyáltalán. De a gyorsan mozgó tárgyak esetén ez nem elégséges.”

Egy másik kényes terület a Crystal Space megvilágításkezelése. Bár lágy árnyékokkal támogatja a színes statikus és dinamikus megvilágítást, nagyon nehéz rávenni, hogy mindezeket kellő sebességgel valósítsa meg a legkülönfélébb játékhelyzetekben. Ezen a területen a grafikus motor még kihívásokkal küszködik.



5. kép Környezet, melyet a Crystal Space tájleképezője hozott létre



6. kép A fényes felületre hulló hó a Crystal Space visszatükröző képességét szemlélteti

A Crystal Space jövője

A Crystal Space-közösség számára nyilvánvalóan elkél a programozók segítsége. Tyberghein olyanokat keres, akik képzettek a grafikus motorok magjának programozásában és járatosak az algoritmikus gondolkodásban – főleg olyanokat vár, akik segítenek a motormag teljesítményének finomhangolásában. „Sokan segítenek a Crystal Space egyéb területein (például: OpenGL- és Direct3D-programozás, windowsos és linuxos átültetések), de csak nagyon kevesen képesek magának a grafikus motornak a fejlesztésében közreműködni”– mondja. „Ha több jó programozónk lenne, akkor sokkal többre lennénk képesek” – állítja Zabolotny. „Elsősorban olyanokra lenne szükségünk, akik járatosak a keresztrendszeres C/C++ programozásban.”

E cikk írásakor a Crystal Space-csapat elsődleges célja az API kellő üzembiztonságának megvalósítása. „A fejlesztői példány már tűrhetően üzembiztos, ám akad még teendőnk bőven” – mondja Tyberghein. A Crystal Space jelenlegi 0.90-es kiadása a régóta várt 1.0-s kiadás elődje. A 0.90-es és az 1.0-s kiadás között az API számottevően nem fog változni, viszont a 0.90-es kiadás célkitűzése, hogy felgyorsítsa a hibakeresést és a leíráskészítést.

A 0.90-es kiadásban kipróbált egyik újdonság a megújult tájleképező motor, amelyet szorosabban és jobban bevontak a Crystal Space-kódjába, mint az előző kiadásokban. A grafikus motor számos új különleges hatást (effect) képes előállítani, mint például ködök, lencsék, továbbá részecskeleképező rendszerrel is felruházták. Technikai szempontból nézve a Crystal Space-segédeszközök modulárisabbak és egyszerűbben elérhetők lettek. Több bővítmény és kódrészlet, amely a korábbi kiadásokban külön függvénykönyvtárakban foglalt helyet, most egyesítve lett.

A végső kérdés, hogy a Crystal Space eljuthat-e arra pontra, amikor már mindazon képességekkel rendelkezik, amelyek lehetővé teszik a kereskedelmi célú játékfejlesztést, valamint elterjedhet-e olyan mértékben, mint a neves 3D grafikus motorok? Maga Tyberghein is kétségekkel küszködik:

7. kép A Crystal Space honlapja

„Ha megfizeted a Quake III-motor használati díját, egy olyan minőségi termékhez jutsz, ami biztosan működik. Ha terméktámogatásra vágysz, ne használj szabad forrású motorokat. Ha azonban úgy érzed, hogy megvagy a terméktámogatás nélkül is, vagy ha drágállod a használati díjat, jól járhatsz egy nyílt forrású motorral.”

Hieber elismeri, hogy „a Crystal Space mérföldekre van a Quake III-tól”, de nem hiszi, hogy ez bárkit is eltántorítana attól, hogy nagyszerű játékokat készítsen motor segítségével, hiszen a Crystal Space jól átgondolt és tervezett motor, bár nem szükségképpen áll mögötte olyan erőteljes technológia, amely igazán jó minőségű megjelenítést tenne lehetővé. „Nézd meg a Tomb Raidert vagy a Half-Life-ot” – mondja. „Egyikük sem rendelkezik kiemelkedően jó 3D-s motorral, mégis mindkettő sikeres, egyszerűen azért, mert színvonalas játékok.”

A Crystal Space a ➲ http://crystal.sourceforge.net weboldalon kereshető fel.

Linux Journal május, 97. szám

Howard Wen
tíz éve foglalkozik a videojátékiparral. Írásai különböző kiadványokban és olyan webhelyeken jelennek meg, mint a Wired, a Salon.com, a Playboy.com, a GameSpot.com, O'Reilly Network és a Dallas Observer. Kapcsolatba léphetünk vele a honlapján: ➲ http://www.howardwen.com

Borúlátás vagy valóság?

Néhány gondolat a trösztellenes törvényhozásról és a Linux kétséges jövőjéről.

Akik úgy gondolták, hogy az útonállók korszaka már véget ért, nem figyeltek eléggé az elmúlt időszakban. Úgy tűnik ugyanis, hogy a Microsoft a programfejlesztés mellett ezentúl politikával is foglalkozik. Mindent és mindenkit maguk alá gyűrtek, elnyomtak, amikor pedig már nem tudták folytatni, akkor megvásárolták versenytársaikat. Most kitalálták, hogy ugyanezekkel a módszerekkel játszanak majd politikai körökben is, így évi 150 ezer dolláros politikai támogatási költségvetésüket 6,1 millióra növelték, nem kis hátszelet szerezve ezzel maguknak. Utálom ezt a szabványdumát, de több száz millió dollárt költöttek el az amerikai adófizetők pénzéből arra, hogy a trösztellenes eljárás keretében tovább erősítsék a Microsoft uralmát a programozói ipar felett.
Természetesen a Microsoft minden követ megmozgat. Programokat „ajándékoznak" az iskoláknak. Ha az iskolák ingyenesen akarják a Microsoft-termékeket megszerezni, akkor csupán egy egyszerű követelmény kerül az árcédulára: más operációs rendszerek használatát nem taníthatják – gyermekeink tehát hűségnyilatkozatot tesznek a Microsoftnak. És ez nemcsak az Amerikai Egyesült Államokban megy így, a Microsoft mindenhol az ajtórésbe nyomja a lábát. Teljesen biztos, hogy ugyanilyen szerződéseket írattak alá kanadai, ausztrál és dél-afrikai iskolákkal is. A Microsoft azt a célt tűzte ki maga elé, hogy legfeljebb öt éven belül az övé lesz az egyetlen operációs rendszer a világon. A hihetetlenül drága és ugyanennyire eredménytelen antitröszt per után az iparágban kiépített állásaik az elkövetkező években erősebbek lesznek, mint legszörnyűbb rémálmainkban elképzeltük. Gyermekeink kizárólag Microsoft-programok és operációs rendszerek használatát fogják tanulni. Az OEM-gyártók csakis Microsoft-termékeket fognak ajánlani, mert mindenki ezeket fogja náluk keresni. Ha tízezer Microsoft operációs rendszer mellé egyetlen Linuxot rendelnek, akkor senki sem fog erőforrásokat ölni egy másik operációs rendszer előtelepítésébe – márpedig amíg a bíróságok nem kötelezik arra a gyártókat, hogy operációs rendszer nélküli gépeket is kínáljanak, addig nem fognak ilyeneket árulni.
Már látom, hogy további értelmetlen perek indulnak. Az eredmény? A Microsoft ellen folytatott amerikai antitröszt per nyerteseinek elsősorban a jogászokat érzem, másodsorban pedig magát a Microsoftot. Mindenki más csak veszít – hacsak nem vásárolt be Microsoft-részvényekből. Újabb tíz év, és a Linux csak múló emlék marad. Borúlátó vagyok? Nem, ez a valóság. Az amerikai bíróságok és tisztviselők rosszul végzik munkájukat. Amíg nem sikerül újabb – harmadik – antitröszt pert nyerni a Microsoft ellen, csak romlani fog a helyzet. Talán akkor – ha egyáltalán eljön az az idő – megtörténik a csoda, és az újítások és a választás szabadsága visszatérhet a programiparba. Nem tartom vissza a lélegzetemet. A jófiúk ugyanis túl sokszor végzik a sor végén.

Axel

A segédprogram állítólag gyorsítja a letöltéseket. Nem mennék bele, hogy letöltéseket lehetséges-e egyáltalán gyorsítani, a program mindenesetre egy vagy több szálat tud indítani, vagyis több kapcsolatot is létre tud hozni a kiszolgálóval. Azt, hogy a letöltések gyorsabbak vagy sem, és melyik mód az előnyösebb, neked kell eldöntened. Én csak annyit tudok mondani, hogy a program sokkal kisebb, mint a többi hasonló alkalmazás. A futtatáshoz libpthread és glibc szükséges.
➲ http://www.lintux.cx/axel.html

nmbscan

Ha a hálózatodban vegyesen vannak Windows és Linux/Unix operációs rendszert futtató gépek, a program segítségével röviden áttekintheted a helyi hálózatot. Egyszerű parancsfájl, amely a rendszeren található hálózati eszközök segítségével azonosítja a windowsos munkaállomásokat, a Samba-kiszolgálókat, és a lehető legtöbb tájékoztatást adja a windowsos gépekről. A futtatáshoz szükséges: `/bin/sh`, smbclient, nmblookup, arp, host, ping.
➲ http://gbarbier.free.fr/prj/dev/#nmbscan

myPhile

Rendkívül egyszerű, mégis sokoldalú felület bármely MySQL-adatbázishoz, amelyhez csatlakozni szeretnél. A telepítés után egy kisméretű adatbázis-telefonkönyvet hoz létre. Könnyedén átírható az egyéni céloknak megfelelően, és apróbb módosításokkal kisebb vállalkozások is szinte képzés nélkül használhatják. Nagyszerű választás, ha például gyors és egyszerű telefonkönyvet kell létrehozni a munkatársak számára. Futtatásához webkiszolgáló PHP-támogatással és webböngésző szükséges.
➲ http://www.rni.net/~geoffm

Jmol

Ha kémiát is tanulsz a főiskolán, és molekulák modellezése is szerepel a feladataid között, ne hagyd ki. Emlékszem, jó néhány évvel ezelőtt hasonló modellezést kellett csinálnom, de akkoriban még külön modellkészletet kellett hozzá használnom – elég mókás volt. A program segítségével a modellek mindenféle barkácsolás nélkül elkészíthetők. Számos példa jár hozzá, de a Weben is sokféle vegyületet találhatsz. Az alkalmazás számos különböző formátumú vegyületleíró fájlt ismer. A futtatáshoz mindössze Java szükséges.
➲ http://www.openscience.org/jmol

Ennyit erre a hónapra.

Linux Journal május, 97. 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.