Beköszöntő

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

Ebben a hónapban talán túl jól is sikerült! Mármint hogy az a célunk, hogy mind a kezdőknek, mind a már gyakorlott rendszergazdáknak, programozóknak érdekes cikkekkel kedveskedjünk.
Az újság egyik fele szinte nyers szakszöveg: rendszermaggal, időzítőkkel, megszakításkezelőkkel, ütemezőkkel, nyomtatókapukkal foglalkozik, míg a másik felében olvasóink tovább ismerkedhetnek a rendszer alapjaival, a héjprogramozással, a SuSE- és a Debian-változatok használatával. Sajnos ezért cserébe kevesebbet foglalkozunk a hálózatokkal, inkább a rendszer „mélyebb" részeire összpontosítunk.
Szeretnénk pontosabban megismerni olvasóink véleményét, hogy mely cikkek tetszettek a legjobban, hol javítsunk a tartalmon, mi volt hasznos olvasmány, mi időrabló, van-e értelme a lemezmellékletnek? Sőt azt, szeretnénk, ha olvasóink részt vennének a lap szerkesztésében is, véleményükkel, javaslataikkal segítenék, hogy az Önök igénye szerinti magazinná váljunk! A ➲ www.linuxvilag.hu/kerdoiv címen elérhető egy kérdőív, amellyel az a célunk, hogy bárki elmondhassa véleményét, beleszólhasson a szerkesztőség munkájába. Kérem, olvassák el, töltsék ki ezt a kérdőívet!
Térjünk csak vissza még egy szószszenet erejéig e havi cikkeinkhez! Pontosabban a 46. oldalon kezdődő „Személyi videófelvevő építése" címűhöz, ami ***Christian A. Herzog*** tollából származik. Hogy miért tartom különösen érdekesnek ezt az írást? Mert egy olyan lehetőséget ír le, ami a Linux mindennapi használhatóságát bizonyítja. Maholnap már idehaza is elérhető a DVD-író, főleg, ha arra gondolunk, hogy egy Linux egy DVD-íróval lényegesen magasabb minőségű szolgáltatást képes nyújtani, mint egy sokkal többe kerülő „videolemez-lejátszó és -felvevő" doboz. Ráérő érdeklődők ma már egyetlen számítógépből álló házimozirendszert is gyárthatnak maguknak. Még azokon a bosszantó ventilátorokon kell dolgozni, és a feleségek is megkedvelik kedvenc pingvinünket!

Programvadászat

Tűzróka

A Mozilla-projekt ismét meglepetést okozott: az eddig FireBird (Tűzmadár) névre hallgató böngészőprogram a 0.8-as változathoz érve FireFox (Tűzróka) névre kereszteltetett át, mivel ilyen nevű program már létezett. Mint azt a ➲ http://firebird.sourceforge.net/ oldalon is olvashatjuk, a FireBird adatbázis-kezelő csapata üdvözli ezt a lépést. Ez a program használhatóságát természetesen nem befolyásolja, szerintem ez a jelenlegi böngésző-programok királya!

Szinte minden böngészőt kipróbáltam már, de egyik sem tetszett annyira, mint a FireFox. Debian rendszereken ugyan csak a 0.7-es változat érhető el, emiatt sohasem a ***.deb*** csomagot telepítem, hanem letöltöm az internetről a tar.gz fájlt, és azt használom – így tettem a 0.8-as változat megjelenésekor is. Mindenkinek csak ajánlani tudom ezt a módszert.

A telepítés a következőkből áll: letöltöm, kicsomagolom, futtatom – tehát nem egy ördöngösség. Ha minden felhasználó számára elérhetővé szeretnénk tenni a rendszeren, akkor a ***/opt/FireFox*** könyvtárat ajánlom a telepítéshez. Régi szokásokhoz híven a ***/opt*** könyvtárba mindenféle „külső” (tehát a rendszernek részét nem képező) programokat ajánlott telepíteni. Nekem ebben a könyvtárban

vannak a legfrissebb böngészőprogramjaim, a legutóbbi OpenOffice.org és egy-két olyan program, amit ***.deb*** csomagban nem találtam.

KOffice

E havi korongunkon a KDE-felülethez szorosan kapcsolódó irodai alkalmazásokat összefogó programcsomag a legnagyobb falat. Szövegszerkesztő (KWord), táblázatkezelő (KCalc),

bemutatókészítő (KPresenter) és minden egyéb olyan megtalálható benne, ami az irodai munkavégzéshez elengedhetetlen. Remélhetőleg mindenki megtalálja a megfelelőt a rendszeréhez. Felkerült Connectiva 9, Red Hat Fedora, SuSE 8.0, 8.2, 9.0, Slackware 8.1, 9.0, 9.1 és forráskódváltozatban is. Telepíteni forráskódból a legbonyolultabb, de korántsem lehetetlen, amennyiben nincs más választásunk. Barátságos felület, nagyon jól együttműködik például a Konquerorral. Képes előnézeti képek készítésére, amennyiben az egérmutatóval a KOffice-hoz rendelt dokumentum felett állunk meg, legyen az akár szöveg, akár táblázat vagy bemutató. Az újítások között szerepel, hogy a KWord az OpenOffice.org 1.1-es változatának a formátumát is képes kezelni. Én régebben nem szerettem a KOffice Workspace-felületet, ami minden ide tartozó programot felölel, most azonban sikerült megkedvelnem, azt az egy dolgot leszámítva, hogy sok helyet foglal el a képernyő hasznos felületéből.

Rendszermag

A legújabb 2.6.2-es rendszermag is helyet kapott a korongon. Fordításához, használatához a régebbi rendszereken valószínűleg szükség lesz a hozzá tartozó programok frissítésére. Miután a kernel-image-2.6.2-1-686 rendszermagot telepítettem, frissítenem kellett a `modconf` programot is, mert a régi még nem volt rá felkészítve; így esett, hogy Woodyból SID lett.

Videó

A 40. oldalon kezdődő „Személyi videófelvevő készítése” című cikkünkhöz tartozó programok és kiegészítők nagy része megtalálható a Magazin/Video könyvtárban. Olyan csemegéket találhatunk itt – a fontossági sorrendet most mellőzve –, mint játékok (vdr-games), időjárás-jelentő (vdr-weather), teletext-program (vdr-teletext), VCD-, valamint DVD-lejátszó (vdr-vcd, vcd-dvd).

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

Java 1.5

A Sun Microsystems bejelentette a Java 2 legújabb változatát, pontosabban a Standard Edition (Tiger kódnevű) 1.5-ös próbakiadását, amely a Java programozási nyelv legnagyobb frissítését testesíti meg az 1.4-es változat két évvel ezelőtti megjelenése óta. Az új kiadással bővült a nyelv által kínált lehetőségek köre, a fejlesztők nagyobb teljesítményű, könnyebben felügyelhető, gyorsabban induló, kevesebb erőforrást igényelő, a multimédiát jobban támogató alkalmazásokat készíthetnek – és többek között a felsorolt típusok és a generikus programrészek használatának régen várt lehetősége is elérhetővé vált.
➲ http://java.sun.com/j2se/1.5.0/index.jsp

Az első számú közellenség

Ha a minden idők leggyorsabban terjedő számítógépes férgét kell kiválasztani, a MyDoom mindenképpen dobogós helyre jut. A január végén kétféle, A és B jelű változatban megjelent féreg feladata két cég weblapjának elosztott szolgáltatásmegtagadási támadással történő kifektetése volt: az A változat a SCO, a B pedig a Microsoft ellen irányult.
A MyDoom érdekessége, hogy klasszikus értelemben véve nem nevezhető vírusnak, hiszen semmilyen programhibát nem használ ki – terjedését az elektronikus levélben kapott mellékleteket mérlegelés nélkül megnyitó felhasználók hiszékenysége alapozta meg. A futtatás után saját SMTP-kiszolgálóján keresztül önmagát terjesztő férget működésének csúcsán egyetlen nap alatt 4,5 millió példányban fogták el, amivel a férges levelek a világ levélforgalmának egyhatodát tették ki.
A MyDoom február 1 és 12 között támadja a ➲ http://sco.com oldalt – tevékenységének február elején azonnal meg is lett az eredménye, a SCO weblapját kénytelen volt a ➲ http://www.thescogroup.com cím alá menekíteni, míg az e téren amúgy is komoly erőforrásokkal rendelkező Microsoft oldalát az egyébként kevésbé elterjedt B változat által indított támadás nem tudta megbéntani.
A SCO és a Microsoft 250–250 ezer dolláros jutalmat kínál annak, aki segít megtalálni a MyDoom – a feltételezések szerint Oroszországban élő – készítőjét.

Egyre feljebb

Az IBM és a SuSE Linux együttműködésének eredményeképpen az IBM eServer sorozatának tagjain és az Opteron alapú gépeken futó, 3-as szervizcsomaggal ellátott SuSE LINUX Enterprise Server 8 elnyerte a Common Criteria szerinti CAPP/EAL3+ minősítést (CAPP, Controlled Access Protection Profile, ellenőrzött hozzáférési profil).
Az Egyesült Államok Védelmi Minisztériuma és hadserege által megkívánt minősítések megszerzése nemcsak a védelmi beszerzések felé nyitja meg az utat a Linuxnak, de a kereskedelmi felhasználók számára is fontos üzenetet hordoz a SuSE és az IBM termékeinek megbízhatóságát illetően.
A két cég tavaly augusztusban EAL2+ minősítést szerzett, azóta az operációs rendszert egy biztonsági naplózó alrendszerrel látták el, illetve további szigorú próbákat végeztek rajta – ennek köszönhető a mostani előrelépés. A SuSE Linux egyben COE (Common Operating Environment) minősítést is nyert, ennek meglétét ugyancsak az Egyesült Államok Védelmi Minisztériuma követeli meg a kereskedelmi termékektől; előírásai a szolgáltatáskészletre és az együttműködési lehetőségekre vonatkoznak. Mindezzel a SuSE Linux lett az első olyan Linux-változat, amely COE és CC minősítést egyaránt kapott, és ha a két cég tervei megvalósulnak, akkor rendszereik még komolyabb minősítési eljárásokon is sikerrel fognak részt venni.

Valós időben 64 biten

A Concurrent Computer Corporation bejelentette RedHawk nevű valós idejű Linux-változatának 2.0-s, kifejezetten AMD Opteron processzorokra készített változatát. A 2.6-os rendszermagra épülő, a Red Hat vállalati terjesztéseivel együttműködni képes RedHawk az első valós idejű Linux az AMD legújabb processzoraira. A RedHawk Linux különféle alkalmazásokban már bizonyított, a cég Intel Xeon processzorokra épülő rendszereit repülőgépszimulátorokban, távközlési rendszerek kipróbálására és ipari vezérlésekben használják.
➲ http://www.ccur.com

coLinux – játékszer?

Elkészült a coLinux, a Linux Windows alatti futtatását szolgáló program első változata. Érdekessége, hogy – leírása szerint – nem virtuális gépet hoz létre a gazda operációs rendszer alatt, hanem különleges illesztőprogramként működik, amely lehetővé teszi a Linux-rendszermag futtatását.

A gazda és a vendég operációs rendszer felváltva használja és egyaránt közvetlenül éri el a gép processzorát, így a vendég is közel olyan sebességgel futhat, mintha egyedül használná a gépet. Mivel a coLinux a normál Linux-rendszerekével megegyező bináris formátumot használ, az alkalmazások módosítás nélkül futtathatók alatta.
➲ http://www.colinux.org

Megújuló OpenView

A HP hamarosan Linuxra is átülteti OpenView Operations alkalmazását, miközben megújítja közel tízéves kódra épülő OpenView for Unix rendszerét. Az első modulok pontosan azt a területet fogják megcélozni, ahol a Linux a legerősebb – és ez a biztonsági felügyelet. Az OpenView már most is képes a linuxos gépek felügyeletére, ám maga a felügyeleti konzol csak HP/UX, Solaris vagy Windows-rendszeren használható. A linuxos változat, bár közeli rokonságban áll majd vele, az eltérő API-k és protokollok miatt nem a unixos kiadás újrafordítása lesz. A HP várakozásai szerint a Linux egyre nagyobb szerephez jut majd felügyeleti alaprendszerként, bár az, hogy a Unixot képes lesz-e valaha is teljesen kiszorítani, nehéz megjósolni. A HP ezzel párhuzamosan egy ügynök nélküli felügyeleti rendszert is fejleszt, amely a jelenlegiektől eltérően semmilyen alkalmazás telepítését nem igényli majd a felügyelendő gépre – mindennek a megvalósítását az teszi lehetővé, hogy szabványos felületeken keresztül az operációs rendszerek egyre több távolról használható szolgáltatást és adatot nyújtanak; ezek segítségével egyre kevésbé okoz gondot a távoli kezelés.
➲http://www.openview.hp.com

O'Neill villanykabát

Az O'Neill Europe és az Infineon együttműködésének eredményeképpen a ruházati cég 2004/2005-ös téli kollekciójában megjelenik az első elektronikus ruha: egy hódeszkásoknak szánt dzseki.

A dzsekibe egy lapkát építenek, amelynek a tulajdonosa Bluetooth-kapcsolaton keresztül igénybe vehető telefonálási és MP3-lejátszási lehetőségre tesz szert. A „Hub" nevű kabát anyagába vezető tulajdonságú szálakat szőnek, ezek adják a lapka kapcsolatát a szintén szövetből készült billentyűzettel, a bukósisakba épített hangszórókkal, valamint a telefonálásához a gallérba kerülő mikrofonnal.
➲ http://www.oneilleurope.com

Nemzedékváltás az Intelnél

Az Intel megkezdte legújabb Pentium 4 processzorainak forgalmazását.

A Prescott kódnévvel fejlesztett lapkák immár 90 nanométeres eljárással készülnek, és elődjeikhez képes több felépítésbeli változáson is keresztülmentek.
A leginkább szembetűnő talán az, hogy az új lapkák már 1 MB másodszintű gyorstárral rendelkeznek, de nem szabad elfeledkezni a 13 új utasításról és a továbbfejlesztett Hyper-Threading megoldásról sem.
A Prescott lapkák bemutatásával az Intel egy újabb Pentium 4 Extreme Edition lapkát is bejelentett, a 3,4 GHz órajelű, 2 MB harmadszintű gyorsítótárral ellátott processzor feladata a csúcskategóriás gépre vágyó játékosok és egyéb profi felhasználók szívének a meghódítása.
Az új processzorok az Intel 865 és 875-ös sorozatú lapkakészleteivel használhatók, az áruk 178 dollárról indul.
➲ http://www.intel.com

Jó a térlátásod?

A Sun kísérleti, különleges szemüveg nélkül is használható háromdimenziós felhasználói felületet mutatott be.
A 3D-felület Java alapú, nyílt forrású, jelenleg Linuxon, X fölött fut. Segítségével a felhasználók egészen újszerű módon használhatják a számítógépüket: a háromdimenziós térben egymás elé vagy mögé helyezhetik az objektumokat, oldalra tologathatják vagy több rétegbe szervezhetik őket.
A Sun egyelőre hangsúlyozottan csak kísérletezik; a felület végleges megjelenése, fejlesztési ütemterve még nem készült el, bizonyos szolgáltatásai ugyanakkor a Java Desktop Systemben egy éven belül megjelenhetnek.
A Project Looking Glass grafikus rendszere nem igényli a hagyományos alkalmazások újraírását, azok továbbra is használhatják a megszokott 2D-megjelenítést. Futtatása természetesen viszonylag komoly erőforrásokat kíván meg: a Sun legalább 850 MHz-es processzort, 256 MB memóriát és megfelelő VGA-kártyát javasolt – igaz, ezeket a követelményeket ma már nem nehéz teljesíteni.
A fejlesztés eredményeképpen a linuxos rendszerek teljesen újfajta felületet kaphatnak – kérdés, hogy lesz-e rá igény. A jelenlegi felületek kétféle elgondolást követnek: vagy a Microsoft rendszereit próbálják meg utánozni, vagy valami teljesen mást próbálnak megvalósítani. Mivel a felhasználókat roppant nehéz valami újnak a használatára rávenni, illetve megértetni velük, hogy a dolgaikat teljesen újfajta módon rendezhetik és szervezhetik, az utánozás mindenképpen fájdalommentesebb eljárás – csakhogy a Sun embereinek ez nem volt ínyükre. Munkájuk sikerét majd a jövő és a felhasználók döntik el.
➲ http://wwws.sun.com/software/looking_glass/

Internetezés repülőről

A Boeing egyik leányvállalata, a Connexion by Boeing hamarosan széles sávú internetelérással szeretné ellátni a repülőjáratokat. A szolgáltatással természetesen elsősorban a vállalati felhasználókat célozzák meg, akik az internethasználati lehetőségnek köszönhetően cégük belső hálózatához vagy a világhálóhoz hozzáférve az utazás ideje alatt is akadálytalanul dolgozhatnak. A hálózati hozzáféréshez földi és műholdas csatornákat fognak igénybe venni, a lefedettség már a szolgáltatás indulásakor is meglehetősen jó lesz, Észak-Amerika, az Atlanti-óceán, Európa nagy része, Skandinávia, Oroszország, a Földközi-tenger, Kína, India és Japán felett kezdettől fogva lehet majd internetezni, a további területek ellátását későbbre tervezik.
A szolgáltatást vezeték nélküli hálózati kártyával felszerelt készülékekkel lehet majd igénybe venni, illetve egyes légitársaságok vezetékes ethernetkapcsolatot is fognak kínálni. Ára repülésenként 25 dollár lesz, az utasok lefelé 20 Mbit, felfelé pedig körülbelül 8 Mbit sávszélességen osztozhatnak.
Alkalmazási területként természetesen nem csak az utasoknak kínált szolgáltatások bővítése vetődhet fel. Ha például fedélzeti internetkamerákat szerelnek fel, akkor géprablás esetén nyomon lehet követni a gépen történteket. Egy további alkalmazás a fedélzeten rosszul lévő utasok távoli diagnosztizálása, amelynek eredménye alapján eldönthető, hogy szükség van-e az azonnali – költséges – leszállásra, vagy a gép folytathatja az útját a célállomás felé. További érdekes lehetőségek mutatkoznak a repülőgépek távoli felügyelete, a csatlakozó járatokra szóló menet közbeni helyfoglalás vagy éppen a csomagok kezelése terén.
➲ http://www.connexionbyboeing.com

A sokadik védvonal

A CyberGuard legújabb, PCI635 jelzésű tűzfalkártyája már akár a helyi hálózaton belülről érkező támadások ellen is képes megvédeni használójának a számítógépét. A PCI foglalatba illeszkedő kártya formájában megtestesülő védelmi eszköz a CyberGuard által nemrég felvásárolt SnapGear PC630-as jelzésű, hasonló termékének a megújított változata. Míg magán a kártyán a SnapGear Embedded Linux-változat fut, addig a gépen gazdarendszerként Linux, Windows 2000, Windows XP vagy Windows Server 2003 üzemelhet.
A kártya dinamikus állapotalapú tűzfalat húz a gép elé, támogatja a VPN protokollokat, betörésérzékelő rendszere a Snortra épül, felügyelete pedig webböngészőn keresztül vagy központi kiszolgálóról végezhető el.
➲ http://www.cyberguard.com

Mozgásba lendülünk

Megjelent a Mobility nevű, német fejlesztésű forgalomszimulációs játékprogram 2.0-s változata. A játék valójában sokkal több annál, mint aminek látszik: motorja valós forgalmi

viselkedést utánoz, ahol a játékos döntései a valós élethez hasonló módon alakítanak ki vagy éppen szüntetnek meg például dugókat. A program forgalmi modellje a Weimari Bauhaus Egyetem és a DaimlerChrysler kutatóinak munkája nyomán jött létre.
A Mobility 2.0 Windows és Linux alá tölthető le, kipróbálása 15 napon keresztül ingyenes, utána 14 dolláros áron vásárolható meg.
➲ http://www.mobility-online.de

A Lycoris tenyérbe mászik

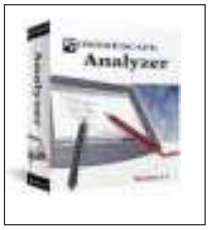
A Lycoris bejelentette, hogy ez év második negyedévében megjelenik tenyérgépekre tervezett operációs rendszere. A Desktop/LX Pocket PC Edition, röviden DL-PPC ígéretük szerint akár vállalati használatra is alkalmas, mégis felhasználóbarát rendszer lesz. Eleinte csak ARM processzorokon fog futni, ami nem nagy hátrány, hiszen ilyen lapka található a Sharp Zaurusban és a HP iPaq gépekben is. A rendszer normál – jelenleg 2.4.18-as számot viselő – rendszermagra épül, így más géptípusokra is könnyen át lehet majd ültetni. A DL-PPC a Windows alapú fájl- és nyomtatómegosztásokat is képes lesz kezelni, a hálózatot vezeték nélküli, USB alapú és infravörös kapcsolaton tudja majd elérni. Személyi adatkezelő alkalmazása képes lesz magát a Lycoris és más gyártók asztali operációs rendszereivel összehangolni, illetve az olyan alapprogramok sem hiányoznak majd belőle, mint egy nagy tudású böngésző, kézírásfelismerő vagy médialejátszó.
➲ http://www.lycoris.com

Qt 3.3

A Trolltech elkészült a Qt 3.3-as és a QSA (Qt Script for Applications) 1.1-es változatával. A Trolltech a számozás tekintetében egészen szerényen viselkedik, a Qt új szolgáltatásai, a .NET-keretrendszer, a 64 bites gépek és az IPv6 támogatása más gyártóknál ki tudja, mekkora változatszám-növekedést hozna. Újdonság a GNU C++ fordító Windows alatti támogatása is; ennek köszönhetően a fejlesztők – a Qt keretrendszer és a gcc fordító segítségével – Unix, Linux, Macintosh és Windows alapú rendszerekre ugyanazt a környezetet használva készíthetnek alkalmazásokat.
➲ http://www.trolltech.com

Algoritmusok, wattok

A PowerEscape állítólag az első olyan kereskedelmi termékeket mutatta be, amelyek az algoritmusok energiafogyasztás szempontjából történő elemzését teszik lehetővé.



A PowerEscape Analyser és a PowerEscape Cache elsősorban algoritmusok és számítógépes rendszerek tervezői számára jelenthet nagy segítséget, már amennyiben a végtermék áramfogyasztása szempont a tervezés során. Segítségével egy-egy algoritmus működése többféle memóriakezeléssel és gyorstárszerkezettel is elemezhető; meghatározható, hogy mely programrészeket lehetne hardveres megoldással gyorsítani; illetve felismerhetők azok a pontok, amelyeknél túl sok adat mozgatása rontja a program teljesítményét. Számos alkalmazás nagy mennyiségű adatot dolgoz fel, ezeknél az energiafogyasztás akár kilencven százaléka is a memória-hozzáférésből fakadhat, így a memóriakezelés hatékonyabbá tételével és a gyorsítótárbeli találatok arányának javításával számottevő akkumulátoros üzemidőnövekedés érhető el például a hordozható eszközöknél.
A két program jelenleg csak elemzésre használható, későbbi kiadásaik várhatóan már programkód módosításra is képesek lesznek.
➲ http://www.powerescape.com

Mr. Kontrolaltdel

Szegény ***David Bradley***! Most, hogy nyugdíjba vonul, mindenki arról beszél, hogy ő írta annak idején azt az aprócska programot, amely – amikor a felhasználó lenyomta a CTRL+ALT+DEL billentyűkombinációt – az IBM PC-ket újraindulásra szólította fel. Pedig az IBM-nél 28 és fél évet eltöltő szakembert, aki egyike volt a PC-t megalkotó csoport tagjainak, sok más munkájáért is elismerés illetné. Mint egyszer ő maga gúnyosan megjegyezte, neki csak a kód megírása jutott, azt „híressé” inkább mások tették, amikor elkészítették a Microsoft hírhedten megbízhatatlan korai operációs rendszereit. Bradley visszavonulását követően az Észak-Karolinai Állami Egyetemen fog tanítani.

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

Számos felhasználó előtt nem teljesen világos, hogy miért más rendszergazdaként vagy különleges jogosultságokkal nem rendelkező felhasználóként beállítani egy linuxos munkaállomást.

Linux-index

1. A Sanchez Computer Associates bankoknak szánt, Linux alapú kulcsfolyamat-rendszere várhatóan legkevesebb ennyi százalékkal csökkenti a teljes birtoklási költséget: 50
2. Világszerte ennyi millióan használnak Linuxot: 18
3. Ennyi a felső árhatára a Commercequest programjának, amely elősegíti, hogy a vállalatok megfeleljenek az új Sarbanes-Oxley társasági előírásoknak: 500 000 dollár
4. Az informatikai költségkeretek ennyi százalékát teszi ki a belső személyzet fenntartása: 70
5. A Wi-Fi-hozzáférésipont-eladások éves növekedési aránya 2008-ig: 50%
6. Várhatóan ennyi millió Wi-Fi hozzáférési pontot adnak majd el 2008-ban: 1
7. A Wi-Fi-hozzáférésipont-eladások becsült növekedési aránya 2005-ben: 132%
8. Ennyiedik helyen áll a Linux a Wi-Fi hozzáférési pontok beágyazott operációs rendszerei között: 1
9. A beágyazott Linux-rendszerrel dolgozó fejlesztők ennyi százaléka választotta a saját fejlesztést mint legkedveltebb Linux-forrást, illetve -gyártót az elmúlt két évben: 15
10. Ennyiedik helyen állt a saját fejlesztés a beágyazott rendszerrel dolgozó fejlesztők legkedveltebb „Linux-forrása, illetve -gyártója” listáján az elmúlt két évben: 2
11. A beágyazott Linux-rendszerrel dolgozó fejlesztők ennyi százaléka választotta a Red Hatet mint legkedveltebb Linux-forrást, illetve -gyártót az elmúlt két évben: 17
12. Ennyiedik helyen állt a Red Hat a beágyazott rendszerrel dolgozó fejlesztők legkedveltebb Linux-forrása, illetve -gyártója listáján az elmúlt két évben: 1
13. A beágyazott Linux-rendszerrel dolgozó fejlesztők ennyi százaléka választja a „saját fejlesztést” mint legkedveltebb Linux-forrást, illetve -gyártót az elkövetkezendő két évben: 18
14. Ennyiedik helyen áll a „saját fejlesztés” a beágyazott rendszerrel dolgozó fejlesztők legkedveltebb Linux-forrása, illetve -gyártója listáján az elkövetkezendő két évben: 1
15. A beágyazott Linux rendszerrel dolgozó fejlesztők ennyi százaléka választja a Red Hatet mint legkedveltebb Linux-forrást, illetve -gyártót az elkövetkezendő két évben: 17
16. Ennyiedik helyen áll a Red Hat a beágyazott rendszerrel dolgozó fejlesztők legkedveltebb Linux-forrása, illetve -gyártója listáján az elkövetkezendő két évben: 2
17. A jövőbeni, beágyazott Linuxot alkalmazó programfejlesztések legkevesebb ennyi százaléka lesz „képes hozzájárulni az alapvetően nem kereskedelmi forrásokhoz”: 50
18. Ennyi millió Apache webkiszolgálót vizsgált meg a Netcraft 2003 novemberében: 30,30

Források
1.: Sanchez Computer Associates
2.: Wired
3–6.: Forbes
7–17: ABI Research, Linux Devices
18.: Netcraft

Linux Journal 2004. február, 118. szám

A Linux és a vírusok

Számos felhasználó előtt nem teljesen világos, hogy miért más rendszergazdaként vagy különleges jogosultságokkal nem rendelkező felhasználóként beállítani egy linuxos munkaállomást. Ha felhasználóként használjuk a rendszert, kétszintű védettséget élvezünk. Először is a rendszer megvédi magát, véletlenül nem vagyunk képesek ártalmas dolgokat művelni, például nem tudunk rendszerfájlokat, programkönyvtárakat törölni. Másodszor a rendes felhasználó nem futtathat a teljes rendszerre hatással lévő rosszindulatú programokat.
Az elterjedt legendával ellentétben a linuxos rendszerek nem teljesen ellenállóak a vírusokkal szemben. Sokkal inkább igaz az, hogy egy jól beállított rendszer jelentős ellenállást tanúsíthat velük szemben. A vírusoknak rendszerszintű elérési jogosultságokra van szükségük ahhoz, hogy kifejtsék a hatásukat, például írásjogra a hajlékony- vagy merevlemezhez. Mivel a rendes felhasználók általában nem rendelkeznek ezzel a jogosultsággal, az ilyen felhasználó alkalmazása hatékony védekezés a vírusok ellen.
Mindezek ellenére a Linux-felhasználóknak is számolniuk kell azzal, hogy a saját fájljaik veszélyben vannak, és ahogyan a linuxos csoportmunka-programokban is egyre elterjedtebb lesz a különböző programnyelvek használata (scripting), a veszély egyre növekedni fog. A vírusok ellen a leghatásosabb védelem mit sem változott: ne futtassunk nem megbízható vagy ismeretlen forrásból származó programot, az elektronikus levelekben érkező programokkal szemben meg kiváltképpen legyünk gyanakvóak, hiszen napjainkban ez a vírusok elterjedésének a legjellemzőbb útja.

Chad Robinson

Linux Journal 2004, 118. szám

Rendszermag-fejlesztési hírek

Bryan O'Sullivan kifejlesztette a Netplugot, egy olyan démont, amely azt figyeli, hogy a hálózati kábelek be vannak-e dugva, és azzal jelzi a válaszát, hogy felizzítja vagy lekapcsolja a rendszer hálózati kapcsolatait. Ez hasznos a laptopok és az olyan rendszerek esetében, amelyek sokat vándorolnak vagy egy Beowulf, esetleg más alegység tagjai. A Netplug a már meglévő, *Lennart Poettering* és mások készítette ifplugd projekthez hasonlóan működik. Meglehet, hogy a két projektet a jövőben megpróbálják egybeforrasztani.

A 2.6-os NFS egy kissé másképp működik, mint a 2.4-es, amely a két sorozat közt tényleges összeférhetetlenséget eredményez. A gondot a `#define NFSEXP_CROSSMNT` kifejezés okozza, amely korábban egy kissé pontatlan jelentésű volt, és amelyet a 2.6-test rendszermagban javítottak ki. Ennek visszaváltoztatása egy pontatlanság visszaállítását jelentené, úgyhogy minden jel arra utal, hogy ez továbbra is gond marad.

Az összeférhetetlenség akkor jelentkezik, amikor egy másik gyártó fejléckódja az `affected #define` sort tartalmazza. Egy egyszerű kanyarral megoldható a kérdéses forráskód hibája.

Junio C. Hamano felvitte *Phillip Lougher* SquashFS tömörített fájlrendszerkódját a 2.4-esből a 2.6-test fába. A SquashFS tűnik a jelenleg legígéretesebb linuxos tömörített fájlrendszernek, bár még mindig csak egyszeri írhatóságot ad, és azután csak olvasni lehet.

Nemrégiben a `digsig` programról írtam: ez egy olyan biztonsági modul, amely végrehajtás előtt ellenőrzi a bináris állomány ellenőrzőösszegét, így biztosítja, hogy az adott rendszeren ne futhasson vírusos kód. Nos, úgy tűnik, engem jobban fellelkesített a hír, mint néhány rendszermagfejlesztőt, nevezetesen *Alexander Viro*-t és *Pavel Machek*-et. Kiderült – és a `digsig` fejlesztői is elismerik –, hogy a `digsig` csupán ideiglenes biztonsági módszer, amit a parancsfájl-írogató ifjoncok nagyon hamar meg tudnának kerülni. Eddig sem vettek a rendszermagba efféle ideiglenes módszereket, mert nem nyújtanak biztos megoldást.

A tavaly november elején kiadott 2.6.0-s rendszermag próbaváltozatával (2.6.0-test1) egyidőben *Linus Torvalds* bejelentette a kód befagyasztását: ez azt jelenti, hogy csak azokat a javítófájlokat fogadják el, amelyek az adatsérülés, a rendszer összeomlása vagy más súlyos hiba kiküszöbölésére szolgálnak. Még a kódok rendezettségére irányuló javítófájlokat is elutasítják, ha azok nem változtatnak ténylegesen a kód működésén. Linus szerint az volt a cél, hogy megkezdhessék a hivatalos 2.6.0-s kiadás végső előkészületeit, és határozottan utalt arra, hogy a kiadás után azonnal átadja a 2.6-os fa karbantartási feladatait *Andrew Morton*-nak. Ez változásnak tekinthető a 2.4-es változathoz képest, amikor *Marcelo Tosatti* csak több kiadás után vette át a fa karbantartását.

Még nem lehet tudni, hogy mikor kezdődik el a 2.7-es fa fejlesztése; talán azonnal, amint Andrew megkapja a 2.6-os fát, de annál korábban azonban biztosan nem.

Rusty Russell és *Tim Hockin* együttesen írt egy javítófájlt, amely a Linux alatt támogatott csoportok számát jóval kétszáz fölé emeli. A javítófájl valójában többezer csoport használatát teszi lehetővé, amire időnként sok embernek (például Tim Samba-ügyfeleinek) van szüksége. Linus Torvalds úgy találta, hogy a javítófájl egyik változata rondább volt, mint a másik, de a jelek szerint addig nem zárkózik el a magasabb csoportszámúvá tételétől, amíg nem kap komolyabb hányingert a megvalósítástól.

Ian Pratt és társai (Computer Laboratory Systems Research Group) elkészítették x86-os kiépítésre az Xen virtuális számítógép-monitor első üzembiztos kiadását, és vendég operációs rendszerként átültették rá a 2.4.22-es rendszermagot. Az Xen egyetlen számítógépen egyidejűleg több operációs rendszert tud futtatni. Amint egy rendszermagot átültetnek az Xenre, a felhasználói bináris fájlok látszólag változatlanul futhatnak, vagyis elméletileg bármilyen Linux-változat képes futni Xen alatt, mindössze egy rendszermag-beültetésre van szükség. Az Xen pillanatnyilag nem futtatható rekurzívan, habár sok elfajzott rendszermagbetyár érdeklődött ez iránt.

Eli Billauer megalkotta a */dev/frandom* véletlenszám-generátort, amely egyes vizsgálatok szerint ötvenszer gyorsabb, mint a */dev/urandom*. Habár titkosításhoz nem eléggé véletlenszerű (ez egyelőre nem is volt cél, de a jövőben talán eléri ezt a szintet), azért jól használható tudományos szimuláció céljára, algoritmusok terhelési próbájához és kódtörés (datawipes) végrehajtásához. Számos jó minősítése ellenére valószínűleg még jó ideig külső javítófájl marad, mivel több rendszermagfejlesztő – köztük *Nick Piggin* – szerint semmi nem indokolja, hogy a */dev/frandom* a rendszermag területén fusson, teljes egészében felhasználói területen is remekül végrehajtható.

2003 október közepén *Roman Zippel* bejelentette az iSCSI szabvány megvalósítását. Noha a megvalósítás nem teljes, és a SysFS helyett az elavult */proc* fájlrendszert használja illesztőfelületként, bizonyos mértékig már hasznavehetőnek tűnik. Roman megígérte, hogy folytatja a kód karbantartását, ha elegendő igény mutatkozik, de ha nem kap segítséget – szavaival élve – ez csak lassan fog menni.

Zack Brown

Linux Journal 2004. február, 118. szám

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

A hónap szakmai tanácsai

Működő USB-s botkormányt!

Egy Logitech Wingman Force 3D erő-visszacsatolásos USB botkormány van a birtokomban. 2.4.20-20.9-es rendszermagot használok (Red Hat 9), amelyben az USB-támogatás engedélyezve van. Van-e mód arra, hogy a botkormányom működjön Linux alatt? A Google segítségével találtam néhány lehetséges megoldást, de nekem egyik sem működött.
Pierre Rochefort, prochefort@rogers.com

Az USB botkormányok jól használhatók Linux alatt. A beállítás mikéntjének megismeréséhez érdemes elolvasni a Linux USB kézikönyvet a ➲ http://www.linux-usb.org oldalon. Ha ennek ellenére is maradnak kérdéseid az USB-támogatás kapcsán, a ➲ linux-usb-users levelezőlistán fel tudod tenni.
Greg Kroah-Hartman, greg@kroah.com

Johann Deneux – a ➲ http://user.it.uu.se/~johannd/projects/ff/index.shtml címen elérhető weboldala szerint – erő-visszacsatolásos meghajtóprogramot fejleszt Linuxhoz. A teljes leírás és a letölthető kód megtalálható az oldalon, így ki lehet próbálni, hogy a te eszközöddel működik-e. Valószínűleg szükség lesz némi kódfordítási és modultelepítési tapasztalatra, de megéri a tanulást. Kipróbálhatod még a Linux Input Drivert (➲ http://linuxconsole.sourceforge.net/input/hardware.html) vagy a Linux botkormánymeghajtót (➲ http://atrey.karlin.mff.cuni.cz/~vojtech/joystick), amelyek állításuk szerint szintén támogatják a szóban forgó eszközt.
Felipe Barousse Boué, fbarrousse@piensa.com

Wi-Fi beállítása SuSE-n zárt módra

Sikeresen telepítettem a SuSE 9-et a noteszgépemre, egy ThinkPad 600E-re, amelynek van egy Netgear MA401 vezeték nélküli kártyája. A SuSE helyesen érzékelte a kártyámat, a 128 bites titkosítást bekapcsoltam és működik. A gond az, hogy kézzel kell zárt módba állítani a kártyát, amikor az elérési ponttal (access point) szeretnék kapcsolatot tartani. Minden egyes újraindítás után a `su` paranccsal rendszergazdai módba kell váltanom és be kell írnom az `iwconfig wlan0 key xxxxxxxx restricted` parancsot, hogy a kártya működjön. A YaST-tal megfelelően beállítottam a kártyát, beleértve a titkosítókulcsot, de magától mindig nyílt módban indul el. Kifelejtettem valamit? Hogyan érhetem el, hogy az `iwconfig` alapértelmezett módja a zárt legyen?
Kevin Lisciotti, lisciotti@yahoo.com

Készíts egy másolatot a */etc/sysconfig/network/ifcfg.template* fájlról és a beállításaidnak megfelelően módosítsd a benne lévő értékeket. Emellett vess egy pillantást a ➲ http://portal.suse.com/sdb/en/2002/11/wavelan.html oldalra, amely részletesebb tájékoztatást nyújt erről a témáról.
Felipe Barousse Boué, fbarousse@piensa.com

Red Hat 7.2 és az ethernetkártya

Mostanában telepítettem a Red Hat 7.2-es terjesztést (ez az első Linux telepítésem), amihez a Linux Administration for Beginners (Linux-felügyelet kezdőknek) című könyv mellékleteként jutottam hozzá. Amikor a rendszert telepítettem, a könyvben leírtakkal ellentétben nem kaptam hálózati beállító-ablakot. A telepítőnek fel kellett volna ismernie a hálózatot, de nem tette. Ezután nem tudtam beállítani az internetkapcsolatot, ami pedig szükséges lenne ahhoz, hogy a súgóoldalakon lévőknél több útmutatást kapjak a rendszer tanulásához.
Egy otthoni PC-t használok a következő összeállításban: Intel Pentium III 500 MHz-es processzor, 256 MB memória, 20 GB WD Caviar IDE merevlemez és Realtek hálózati csatoló (D-Link kártyával próbáltam ki). Egy kábelmodemhez csatlakozom és nincs helyi hálózatom, bár egy második gép rendszerbe állítását is tervezem.
Úgy néz ki, hogy a Red Hat 7.2 nem ismeri fel az `eth0`-t, bár élő kapcsolat van. Mindenfelé kerestem útmutatást. A legtöbben azt tanácsolták, hogy hajlékonylemezről telepítsem a Realtek meghajtóprogramjait, amelyek már a terjesztésben is megtalálhatók, de mint mondtam, én tényleg új vagyok ezen a területen és nincs támpontom. Azon vagyok, hogy veszek egy Mandrake-et vagy Red Hat 9-et, hátha valamelyiknek használhatóbb telepítője van.
Marcel, marcelnh4@hotmail.com

Anélkül, hogy ismernénk a hibaüzeneteket vagy látnánk néhány naplófájlt, igen nehéz kitalálni, hogy mi is történik a rendszeredben. Rendben van a hálózati kártyád? Megfelelően van telepítve? Nincs valamilyen összeférhetetlensége egy másik eszközzel? Eltérő Linux-változat vásárlása helyett inkább egy másik hálózati kártyát kellene venni, olyat, amit biztosan támogat a Linux. Nézd meg a ➲ http://hardware.redhat.com/hcl oldalt és keress olyan kártyát, ami megfelel a pénztárcádnak és a rendszeredenek.
Felipe Barousse Boué, fbarousse@piensa.com

Az eszközgondok kizárása végett rendszeredet próbáld ki a Knoppixszal (➲ http://www.knoppix.net), ami egy CD-ről futtatható Linux-rendszer. A Knoppixot szabadon le lehet tölteni és lemezre lehet sütni. Ha a Knoppix futtatásakor a kártya működik, akkor már csak idő kell hozzá, hogy némi rendszergazdai tapasztalatot gyűjts, ami egy újabb terjesztésre történő frissítésnél is jól jöhet. Első Linux-változatként próbálj olyat választani, amit a helyi felhasználói csoport vagy más általad használt támogatás ismer.
Don Marti, dmarti@ssc.com

Linux Journal 2004. február, 118. szám

Új termékek

NIOS-felület

A Net Integration Technologies olyan független számítási hálózatot kínál, amely a Linuxon alapuló Net Integrator Operating Systemre (NIOS) épül. Az önjavítónak és önkarbantartónak tervezett NIOS-felület bárhol kapható eszközökre és alkatrészekre épít és a kis, illetve közepes méretű üzleteket célozza meg. Maga a NIOS nyílt forrású programból áll; a mérete 16 MB. A NIOS Platformnak része a NetIntelligence, egy mesterségesintelligencia-modul, amely független szolgáltatásokat használ rendszerösszetevők és a belső alrendszerek létrehozására, telepítésére és karbantartására, beleértve tűzfal- és DHCP-jellemzőket, valamint a DNS-bejegyzéseket. A SystemER program a rendszer helyreállítását egy végzeteshiba-rendszerösszeomlásból két percen belül lehetővé teszi. Egyéb összetevői közé tartozik az Exchangelt! munkacsoportos kiszolgáló, a TunnelVision intelligens VPN-megoldás, amely állandó IP-cím nélkül is működik, az Expression Desktop és a DoubleVision, ami egy olyan redundáns internetkapcsolódási módszer, amit arra terveztek, hogy több nagy sebességű csatolót lehessen NIOS-szal meghajtott kiszolgálókhoz kapcsolni.
➲ http://www.net-itech.com

Xandros Desktop 2.0

A Xandros Desktop 2.0-s egy könnyen használható grafikus környezet, amit mindössze négy egérkattintással lehet telepíteni. A Debian Sarge-terjesztésre és a Xandros bővítéssel ellátott KDE 3.1.4-re épülő rendszer legfontosabb tulajdonságai közé tartozik a nagyon egyszerű telepítés önműködő lemezrészelosztással, a húzd és ejtsd CD-írás a fájlkezelőn belül, valamint a fájl- és erőforrás-megosztás a Windows-hálózatokkal. A 2.0-s munkaasztal kínálatában fellelhető az egy ablakon belül fülekkel elkülönített webböngészés, valamint a felugró ablakok és reklámcsíkok kikapcsolásának lehetősége, egy önműködő levészemétszűréssel ellátott levelezőügyfél, egy beszélgetőügyfél, amely együttműködik az MSN, Yahoo, AOL, ICQ és IRC-szolgáltatásokkal, valamint az OpenOffice.org 1.1. A Xandros Desktop 2.0 Standard és Deluxe kiadásban jelenik meg, a Deluxe változat tartalmazza a Crossover Office 2.1-et, egy 350 oldalas kézikönyvet, extra játékokat, alkalmazásokat és eszközöket, valamint 60 napos, elektronikus levélben igénybe vehető támogatást.
➲ http://www.xandros.com

TimeStorm Linux Tool Suite

A TimeSys új TimeStorm Linux Tool Suit olyan fejlesztőeszközöket tartalmaz, amelyek a teljes beágyazott-Linux fejlesztési ciklust támogatják, függetlenül attól, hogy milyen típusú beágyazott operációs rendszert használunk. A TimeStorm csomag az Eclipse-felületre épül, s vannak eszközei a rendszermag-átültetés kezeléséhez, alkatrészek összevonásához (hardware integration) és a mindenre kiterjedő kipróbálási és véglegesítési igényekhez. A TimeSys TimeStorm összevont feljesztői környezetének bővítményeként (plugin) szállított Tool Suite minden beágyazott Linux-terjesztéssel működik, legyen az házilag vagy gyárilag készített. A Tool csomag magában foglalja a TimeStorm Linux Verification Suite-ot, ami egy olyan keretrendszer, amely önműködővé teszi a teljes beágyazott terjesztésnek és alkalmazásainak a kipróbálását, illetve bevizsgálását a fejlesztés minden egyes szakaszában. Több mint 1150 nyílt forrású próba áll rendelkezésre. A Linux Development Suite-ban található eszközök segítenek a fejlesztőnek egyedi Linux-rendszert építeni és átültetni azt a céleszközre. A Linux Hardware-Assisted Debug (hardverrel segített hibakeresés) segít a hibakeresésben, inicializásában és a Linux indításában oly módon, hogy kapcsolófelületet biztosít a TimeStorm IDE, valamint a JTAG és a GDB-t használó, lapkába épített hibakeresők között.
➲ http://www.timesys.com

MontaVista DevRocket Development Environment

A MontaVista DevRocket 1.0-s egy teljesen egybevont (integrated) grafikus fejlesztői környezet beágyazott Linuxhoz. Az Eclipse technológián alapuló megoldás különböző felületeken, például Linuxon, Microsoft Windowson és Solarison is hasonló külsőt és felületet nyújt a fejlesztéshez. A MontaVista DevRocket a legújabb Eclipse 2.1-en alapul, lehetővé téve a vásárlóknak és az internetszolgáltatás-fejlesztőknek az Eclipse-felület minden előnyének a kiélvezését, belértve az Eclipse-közösség által készített külső fejlesztéseket, csakúgy, mint az Eclipse által támogatott eszközök sokaságát. Az egybevont létfontosságú fejlesztési képességek – például a fordítás, szerkesztés és hibakeresés mellett – a DevRock könnyen használható projektvarázslókat is nyújt, amelyekkel az általános beágyazott fejlesztési tennivalókat önműködővé lehet tenni.
➲ http://www.mvista.com

Global Navigator 5.0

Az NEC Global Navigator 5.0-s egyes vagy többszörös kapcsolati központok hívásaktivitásának és ügynökteljesítményének a megfigyelésére készült. A Global Navigator vállalati szintű hívásközpont-kezelést, valamint az egy vagy több hálózaton elérhető munkahely vezérlését oldja meg. Az alkalmazás az Infocast elnevezésű, valós idejű információs programot használja, amely a kapcsolati központ statisztikáit az ügynök munkahelyének egy kis ablakába küldi. Az új, 5.0-s változat SCO UNIX-ról lett átültetve Linuxra és MySQL-t használ adatbázis-kezelő rendszerként.
➲ http://www.cng.nec.com/cng

Linux Journal 2004, 118. szám

Linux kontra SCO

A rendbontó technológiák elutasítása elhibázott lépés.

Amikor a piacon megjelenik egy rendbontó technológia, a piac régi szereplői – a strucchoz hasonlóan – mindent megtesznek azért, hogy ne vegyenek róla tudomást. A rendbontó technológiák általában vagy erőtlenül indulnak, vagy olyan válaszokként, amelyek keresik a kérdésüket, így könnyű semmibe venni, leírni az okoskodó betolakodókat. Amikor azonban ezeknek a módszereknek követői akadnak és ostrom alá veszik a területet, a hatalom birtokosai ébredésükkor két válasz közül választhatnak: vagy felkarolják a rendbontót, vagy megpróbálják eltaposni.

A Linux iskolapéldája a rendbontó módszereknek. Programozó-tanoncok játékaként, főiskolás feladatként kezdte életútját. Az interneten keresztül gyorsan elterjedt, de még ekkor is jóval elmaradt a hagyományos Unix-rendszerek mögött; képességei korlátokba ütköztek, leírásai hiányosak voltak.

A Linux azért volt rendbontó (és az még ma is), mert működik. A nyílt forrású fejlesztői modell lassan, de biztosan – mint a mesebeli teknőc –, könyörtelenül tolta előre a GNU/Linux operációs rendszert és az őt kiegészítő segédprogramokat, mígnem ezek elérték, majd elébe vágtak a döbbent, szabadalmakkal felvértezett nyulaknak.

A Nyílt Forrás Közösségének szellemisége és a GPL, LGPL, BSD és NLP/MLP felhasználási szerződések által nyújtott biztosítékok segítenek elkerülni a kereskedelmi Unixokra oly jellemző széthúzást, és jogi alapot nyújtanak a nyílt forrású folyamat fejlődéséhez.

A Linuxot felkaroló és eltaposni próbáló erők között tátongó szakadék széles és mély. Az olyan régóta piacvezető cégek, mint az IBM, a HP vagy az Oracle új üzleti lehetőségeket sejtenek a Linux-rendszerekben, annak ellenére vagy talán pontosan azért, mert érdekeltek a kereskedelmi programokban, beleértve a zárt forrású Unixokat is. Az ellenkező tábor legjelentősebb képviselője, a Microsoft ősrégi Unix-gyűlölő, a Linux „természetes ellensége" – utálja mind nyílt származását, mind nyílt forrású fejlesztői modelljét.

A szegény eltévelyedett SCO valahol a szakadék mélyén bolyong. A SCO Group, amely a Linux terjesztésével foglalkozó Calderából született újjá, most kitagadva érzi magát, mert korábban kereskedelmi Unixot vásárló felhasználói tábora Linuxra vált át. Az IBM-mel szemben támasztott követelésekből és a Linux-felhasználók fenyegetéséből látszik, hogy a SCO a Caldera korábbi barátságos ölelésének erejét a fojtogatásig igyekszik fokozni. A lekörözött informatikai cégek régi és sajnálatos szokásukhoz híven a bírósághoz fordulnak, hogy megpróbálják elérni mindazt, amit a szabad piacon nem tudtak. Ha egy cég jogi úttal próbálkozik sikeres üzletpolitika helyett, az minden esetben nem várt következményekhez vezet. Az, hogy a SCO az IBM AIX-eladásokból próbálja pótolni az elvesztett piaci részesedése miatti bevételkiesést, ahhoz fog vezetni, hogy az IBM amúgy is csökkenő kereskedelmi Unix-eladásai még gyérebbé válnak.

Az, hogy a SCO felhasználási díjakat próbál meg kisajtolni a Linux alapú kereskedelmi megoldások szállítóitól és a beágyazott Linuxszal kereskedőktől, lehet, hogy rövid távon lelassítja a Linux terjedését, de befagyasztja a SCO Linuxból remélt bevételeit is. Az önkiábrándítás jellemző válasz a rendbontó módszerekre.A SCO által kiszabott 699 dolláros kereskedelmi és 32 dolláros beágyazott felhasználási díj arról árulkodik, hogy a cég a fellegekben jár. Ez a hercehurca egy cseppet sem javít a SCO Unix helyzetén, és nem fogja ***Darl McBride***-ot Linuxból származó bevétellel elhalmozni. A SCO által elképzelt világban ő maga győzedelmeskedik az IBM elleni perben és a Linux-felhasználási szerződések ügyében, de a cég valójában még mindig vesztésre áll.

A SCO azt képzeli, hogy elfoghatja, megsebezheti vagy akár megölheti a Linuxot? Ez nem fog megtörténni. A feldühödött kereskedelmi és beágyazott Linuxot használók nem fognak egy csapásra SCO Unixot vásárolni, ha az nem nyújt többet a Linuxnál, még a magas rendelkezésre állású rendszerek esetében sem, és beágyazott környezetben teljesen használhatatlan. Az igazán keményvonalasok megmaradnak majd a Linuxnál, és egy olyan változatot hoznak létre, amelyből az összes SCO-maradvány kimarad. E lehetőségek közül egyik sem segít a SCO-n. McBride-nak és a barátainak azt kell megérteniük, hogy a SCO követelései hosszú távon ahhoz vezetnek, hogy a SCO-kód minden előfordulása – legyen szó akár a Linuxban (átmenetileg) található kódról, akár az SCO Unixról – érinthetetlenné és eladhatatlanná válik.

Linux Journal 2004. február, 118. szám

James Ready
A MontaVista Software, Inc. elnöke és ügyvezetője, több mint 25 éves mérnöki és vállalkozói tapasztalattal bír.

A 2.6-os rendszermag új munkasor kezelése

A rendszerteljesítmény egyik központi kérdése a megszakítási lappangási idő. A munkasorok használatával egy meghajtóprogramban kikerülhetjük a megszakításoktól védett időigényes kódrészletek használatát.

A legtöbb Unix-rendszerben – így a Linuxban is – az eszközmeghajtók általában két részre (félre) bontják a meghajtók feldolgozását. Az első részben (a felső félben) található az ismerős megszakításkezelő, amelyet a rendszermag az alkatrésztől kapott megszakítási jel észlelésekor hív meg. Sajnos a megszakítás valóban méltó a nevére: amikor az alkatrész kiadja a megszakítást, az bármilyen futó kódot megszakít. Tehát a megszakításkezelők (a felső felek) aszinkron módon futnak, figyelembe véve a jelenleg futó kódot. Minthogy a megszakításkezelők megszakítják a futó kódot (legyen az akár más rendszermagkód, akár felhasználói folyamat vagy másik megszakításkezelő), nagyon fontos, hogy a lehető leggyorsabban fussanak le.

Még nagyobb baj, hogy néhány megszakításkezelő (Linux alatt gyorsak a megszakításkezelők) a futás során kikapcsolják a megszakításokat a helyi processzoron. Ezt azért teszik, hogy a megszakításkezelő megszakítás nélkül, a lehető leggyorsabban fusson le. Továbbá az összes megszakításkezelő az összes processzoron kikapcsolja a saját megszakítási vonalát. Ez biztosítja, hogy azonos megszakítási vonalnak két megszakításkezelője nem futhat egyszerre. Egyúttal megakadályozza, hogy a meghajtóprogram készítőjének kezelni kelljen a rekurzív megszakításokat, ami bonyolítaná a programozást. Ha azonban a megszakítások le vannak tiltva, más megszakításkezelők sem tudnak futni. A megszakításlappangás (interrupt latency) – vagyis hogy mennyi időbe telik a rendszermagnak, amíg egy alkatrész megszakítási kérelmére válaszol – a teljesítmény egyik kulcskérdése. Ismétlem, a megszakításkezelők sebessége súlyos kérdés. Kisméretű gyors megszakításkezelők írásának az elősegítésére a második részt, avagy az alsó felet használjuk. Ennek a résznek a feladata, hogy a lehető legnagyobb mennyiségű, későbbre halasztható munkát átvegye a felső résztől. Az alsó rész bekapcsolt megszakítások mellett fut, ennek megfelelően az alsó fél futtatása nem akadályozza a többi megszakításkezelő futását – így nincs is hátrányos hatással a megszakítási lappangásra.

Csaknem minden eszközmeghajtó alkalmaz ilyen vagy olyan alsó felet. Az eszközmeghajtó a felső fél (a megszakításkezelő) segítségével válaszol az alkatrésznek, illetve itt végzi el a fontos időérzékeny műveleteket, például az eszközregiszterek alaphelyzetbe történő állítását vagy az eszközből a memóriába történő adatmásolást. A megszakításkezelő ezután megjelöli az alsó felet, utasítva a rendszermagot, hogy a lehető leghamarabb futassa le a szükséges kódot, majd kilép.

A legtöbb esetben a valódi munka csak ekkor kezdődik az alsó félben. Egy későbbi időpontban – gyakorta a megszakításkezelő visszatérése után azonnal – a rendszermag az alsó felet is végrehajtja. Az alsó fél lefut és elvégzi azokat a feladatokat, amelyeket a megszakításkezelő hátrahagyott. Az alsó és felső fél közötti munkamegosztás teljes egészében az eszközmeghajtó-készítő döntésén múlik. Az eszközmeghajtó-készítők a lehető legtöbb munkát általában megpróbálják az alsó részre hárítani.

A Linux – zavaró módon – elég sokfajta módszert ajánl fel az alsó rész megvalósítására. Jelenleg a 2.6-os rendszermag `softirq`-k, `tasklet`-ek és munkasorok (work queues) típusú alsórész-megoldásokkal bír. A korábbi rendszerekben másfajta alsó felek voltak elérhetők: például a BH-k és a feladatsorok (task queues interface). Ez a cikk kizárólag az új munkasorfelülettel (work queue) foglalkozik, amely a 2.5-ös fejlesztői sorozatban mutatkozott be a feladatsorok betegeskedő `keventd` részének a kiváltása végett.

Bevezetés a munkasorok világába

A munkasorok két dolog miatt is érdekesek: először is az összes alsófél-megoldás közül ezeket a legegyszerűbb használni. Másodszor ez az egyetlen olyan alsófél-megoldás, ami folyamatkörnyezetben (process context) is képes futni, így a munkasorok jelentik az egyetlen lehetőséget, amivel az eszközmeghajtó-készítő akkor élhet, ha az alsó résznek aludnia kell. Továbbá a munkasorok képessége vadiúj, és szerintem ami új, az király.

Nézzük meg, mivel is jár az, hogy a munkasorok a folyamatkörnyezetben futnak! Ez jelentős különbség más alsófél-megoldásokhoz képest, amelyek – ettől eltérő módon – megszakítási környezetben futnak. A megszakítási környezetben futó kód nem alhat vagy blokkolhat, minthogy a megszakítási térnek semmilyen újraütemezhető háttérfolyamata nincsen. Ezért, mivel a megszakításkezelők nincsenek folyamathoz rendelve, az ütemezőnek nincs mit alvó állapotba helyeznie, és ami talán még fontosabb: nincs

mit felébresztenie sem. Következésképpen megszakításkörnyezetben nem lehet elvégezni olyan műveleteket, amelyek a futó környezetet a rendszermagban alvó állapotba helyeznék. Ilyen művelet a jelző lekapcsolása (downing a semaphore), a felhasználói tér memóriájába vagy onnan történő másolás, vagy a nem önműködően történő memóriafoglalás. Minthogy a munkasorok folyamatkörnyezetben futnak (mint azt látni fogjuk, a rendszermagszálak hajtják őket végre), teljes mértékben képesek az alvásra. A rendszermag lényegében pontosan úgy ütemezi az alsó felek futását a munkasorokban, mint a rendszer bármely más folyamatát. Akárcsak a többi rendszermagszál (kernel threads), a munkasorok is alhatnak, meghívhatják az ütemezőt és így tovább.

Normál esetben a munkasorokat az alapértelmezett rendszermagszálak kezelik. Processzoronként egy ilyen alapértelmezett szál fut. Ezeket `events/n` alakú névvel jelöljük, amelyben ***n*** annak a processzornak a száma, amelyhez a szál kötődik. Például az egyprocesszoros gépeknek csak egy `events/0` rendszermagszála lenne, míg a kétprocesszoros gépeknek már egy `events/1` nevű szála is lenne. Munkasorainkat azonban akár saját rendszermagszálon is futtathatjuk. Amikor az alsó felünk elindul, az alapértelmezett szál helyett a saját egyedi szálunk ébred fel és kezd el dolgozni. Egyedi munkasorszálat (unique work queue thread) készíteni csak néhány teljesítményfüggő esetben érdemes. Általában az alapértelmezett szálat használó alsó feleket érdemes előnyben részesíteni. Ennek ellenére később azt is meg fogjuk nézni, hogy miként lehet munkasorszálat készíteni.

A munkasorszálak alsó részünket munkasorkezelőnek nevezett különleges függvényként hajtják végre – a munkasorkezelőben készíthetjük el az alsó részünket. A munkasorfelület használata nagyon könnyű; az egyetlen nehéz rész magának az alsó résznek (azaz a munkasorkezelőnek) az elkészítése.

A munkasor illesztőfelülete

A munkasorok használatához először is létre kell hoznunk egy munkasorszerkezetet. A munkasorszerkezetet a ***linux/workqueue.h*** fájlban megadott `struct work_struct` határozza meg – szerencsére a munkasorszerkezetek előállítását két makró könnyíti meg. Amennyiben munkasorszerkezetünket statikusan szeretnénk meghatározni (azaz úgynevezett globális változóként), a következő kifejezéssel közvetlenül megadhatjuk:

```
DECLARE_WORK(name, function, data)
```

Ez a makró létrehozza a `struct work_struct` szerkezetet, majd a megadott munkasorkezelő és függvény felhasználásával alaphelyzetbe állítja. Munkasorkezelőnknek a következő prototípusnak kell megfelelnie:

```
void my_workqueue_handler(void *arg)
```

Az `arg` értéket a rendszermag a munkasorkezelő minden egyes meghívásakor átadja – ezt a `DECLARE_WORKQUEUE()` makróban található `data` érték határozza meg. Az eszközmeghajtók az érték használatával több munkasort is kezelni tudnak egyetlen munkasorkezelő segítségével. A `data` értéket a munkasorok megkülönböztetésére is felhasználhatjuk. Amennyiben munkasorszerkezetünket közvetlen módszer helyett inkább dinamikusan szeretnénk létrehozni, arra is lehetőségünk nyílik. Ha a munkasorszerkezetre csak közvetett hivatkozásunk van (például azért, mert a `kmalloc()` függvénnyel foglaltuk le), a következő módon állíthatjuk alaphelyzetbe:

```
INIT_WORK(p, function, data)
```

Ebben az esetben a `p` a `work_struct` szerkezetre mutat majd, a `function` a munkasorkezelőnk, a `data` pedig az az érték, amit a rendszermag a meghíváskor át fog adni. A munkasorszerkezet létrehozását normál esetben csak egyszer végezzük el, például a meghajtónk alaphelyzetbe állító eljárásában. A rendszermag a munkafolyam-szerkezetet használja a rendszeren futó különféle munkasorok nyomon követésére. Érdemes figyelemmel kísérnünk a szerkezetet, mert később még szükségünk lesz rá.

A saját munkasorkezelőnk

Munkasorkezelőnk lényegében bármit csinálhat, végtére is a saját alsó részünkről van szó! Az egyetlen megszorítás, hogy kielégítse a megfelelő prototípust. Minthogy munkasorkezelőnk folyamatkörnyezetben fut, szükség esetén aludni is tud.

Megvan a munkasor-adatszerkezetünk és a munkasorkezelőnk, de hogyan tudjuk futtatni? Ha azt szeretnénk, hogy az adott munkasorkezelő a legközelebbi, a rendszermag számára is kényelmes helyen fusson le, hívjuk meg a következő függvényt és adjuk át neki a munkasorszerkezetünk címét:

```
int schedule_work(struct work_struct *work)
```

Ha a besorolás sikeres volt, a függvény nullától eltérő értéket ad vissza, hiba esetén azonban nullát. A függvény folyamat- vagy megszakításkörnyezetben egyaránt hívható. Néha előfordul, hogy a munkasort nem azonnal akarjuk beütemezni, csak egy bizonyos idő eltelte után; ilyen esetekben a következő utasítást használjuk:

```
int schedule_delayed_work(struct work_struct *work,
                          unsigned long delay)
```

Ilyenkor az adott munkasorszerkezethez rendelt munkasorkezelő a megadott ideig még biztosan nem fog lefutni. Például ha van egy `my_work` nevű szerkezetünk és öt másodpercre szeretnénk időzíteni, a következő formát használjuk:

```
schedule_delayed_work(&my_work, 5*HZ)
```

Normál esetben munkasorkezelőnket a megszakításkezelőnkből időzítjük, de semmi sem akadályoz meg bennünket, hogy ezt most tetszés szerinti helyről tegyük meg. Normál esetben az alsó fél és a megszakításkezelő csapatként együtt dolgozik, közösen oldják meg az eszközzel kapcsolatos feladatokat. A megszakításkezelő mint a

Munkasor-függvényreferencia

Statikus munkasorszerkezet-készítés:

```
DECLARE_WORK(név, függvény, adat)
```

Dinamikus munkasorszerkezet:

```
INIT_WORK(p, függvény, adat)
```

Új munkaszál készítése:

```
struct workqueue_struct
*create_workqueue(const char *név)
```

Munkaszál törlése:

```
void
destroy_workqueue(struct workqueue_struct *wq)
```

Munka adott munkaszálhoz rendelése:

```
int
queue_work(struct workqueue_struct *wq,
           struct work_struct *work)
```

Munka ütemezése adott idő eltelte után adott munkaszálhoz:

```
int
queue_delayed_work(struct workqueue_struct *wq,
                   struct work_struct *work,
                   unsigned long delay)
```

Függőben lévő munkák kivárása az adott munkaszálon:

```
void
flush_workqueue(struct workqueue_struct *wq)
```

Munka ütemezése az alapértelmezett munkaszálon:

```
int
schedule_work(struct work_struct *work)
```

Munka ütemezése adott idő eltelte után az alapértelmezett munkaszálon:

```
int
schedule_delayed_work(struct work_struct *work,
                      unsigned long delay)
```

Függőben lévő munkák kivárása az alapértelmezett szálon:

```
void
flush_scheduled_work(void)
```

Az adott késleltetett munka leállítása:

```
int
cancel_delayed_work(struct work_struct *work)
```

megoldás felső fele általában előkészíti a megmaradt munkát, majd beütemezi az alsó felet. Az is elképzelhető, hogy a munkasorokat az alsófél-feldolgozáson kívül másra szeretnénk használni.

Munkasorkezelés

Amikor besorolunk egy munkát, az a munkaszál következő ébredésekor fut majd le. Néha előfordul, hogy folytatás előtt biztosítanunk kell a munkasor befejeződését a rendszermagban. Ez különösen a modulok esetében fontos, amelyeknél a modul kivétele előtt az összes alsó félnek be kell fejeződnie. Az igény kielégítésére a rendszermagban egy függvényt találunk, amely megvárja, amíg a függőben lévő munkaszálon az összes munka lefut:

```
void flush_scheduled_work(void)
```

Minthogy ez a függvény a munkaszálon található összes függőben lévő munkafolyamatot kivárja, a dolog viszonylag hosszú időt vehet igénybe. Miközben a függvény a munkaszálak befejezésére vár, a hívás alszik. Következésképpen ezt a függvényt kizárólag folyamatkörnyezetben hívhatjuk meg. Ne használjuk a függvényt, ha ütemezett munkánk befejezését nem feltétlenül kell biztosítanunk. A függvény nem üríti ki a függőben lévő időzített munkákat. Amennyiben időzítéssel ütemezzük a munkát és az időzítés még nem járt le, a munkasor ürítése előtt le kell állítanunk az időzítést:

```
int cancel_delayed_work(struct work_struct *work)
```

Ez a függvény az adott munkasorszerkezethez tartozó időzítést állítja le – más munkasorokra nincs hatással. A `cancel_delayed_work()` függvényt kizárólag folyamatkörnyezetben hívhatjuk meg, mivel alhat. Ha valamilyen munkát leállított, nullától eltérő értéket ad vissza, egyéb esetben a visszaadott érték nulla.

Új munkaszál létrehozása

Ritkán az alapértelmezett munkasorszál nem elég. Szerencsére a munkasor-csatolófelület lehetővé teszi, hogy saját munkaszálat készítsünk és ezt használjuk alsó részünk futtatására. Új munkasorszál létrehozására a következő függvényt használjuk:

```
struct workqueue_struct *
create_workqueue(const char *name)
```

A rendszer elindítása előtt (system initialization) a rendszermag például a következőképpen készíti el az alapértelmezett szálakat:

```
keventd_wq = create_workqueue("events");
```

Ez a függvény készíti el a processzoronkénti összes munkaszálat. Visszaadott értéke a `struct workqueue_struct` szerkezetre hivatkozó mutató, ami a munkasor más munkasoroktól (például az alapértelmezettektől) való megkülönböztetésére szolgál. A munkaszál létrehozása után hasonlóképpen rendelhetünk hozzá munkákat, mint az

alapértelmezett munkaszálakhoz:

```
int queue_work(struct workqueue_struct *wq,
               struct work_struct *work)
```

Itt a `wq` a `create_workqueue()` függvénnyel készített, a munkasort azonosító mutató, a `work` pedig a munkasorszerkezetünkre hivatkozó mutató. Másik megoldásként a munkát időzítve is ütemezhetjük:

```
int
queue_delayed_work(struct workqueue_struct *wq,
                   struct work_struct *work,
                   unsigned long delay)
```

E függvény működése azonos a `queue_work()` függvényével, kivéve, hogy a besorolást a `delay` kapcsolóban megadott pillanatig elhalasztja. A két függvény megfelel a `schedule_work()` és `schedule_delayed_work()` függvényeknek, eltekintve attól, hogy az adott munkát az alapértelmezett helyett a megadott munkasorba helyezik el. Mindkét függvény siker esetén nullától eltérő értéket, hiba esetén pedig nullát ad vissza. Mindkét függvény folyamat- vagy megszakításkörnyezetből egyaránt hívható.
Végül egy adott munkasort a következő utasítással üríthetünk ki:

```
void flush_workqueue(struct workqueue_struct *wq)
```

A függvény addig vár, amíg `wq` munkaszálon az összes ütemezett munka be nem fejeződik, és csak azután tér vissza.

Összegzés

A munkasor-csatolófelület a 2.5.41-es változattól része a rendszermagnak. Akkoriban igen sok meghajtó és alrendszer kezdte el munkahalasztási módszerként használni. Felmerül a kérdés: biztos, hogy ez az eszményi alsó rész számunkra? Amennyiben az alsó részünket folyamatkörnyezetben kell futtatnunk, valóban a munkasor a megfelelő választás. Továbbá ha esetleg rendszermagszálakat szeretnénk létrehozni, valószínűleg úgyszintén a munkasorok alkalmazása lesz a legjobb megoldás. De mit tegyünk, ha nincs szükségünk olyan alsó részre, ami aludni is tud? Ez esetben valószínűleg a `tasklet`-ek használata a jobb választás. Ezeket is könnyű alkalmazni, de nem használnak rendszermagszálakat. Minthogy nem folyamatkörnyezetben futnak, végrehajtásuk során környezetváltási pluszmunka sincs – így számunkra is kevesebb többletmunkát okoznak.

Linux Journal 2003. november, 115. 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 punkzenét hallgat.

Memóriafoglalás a rendszermagban

Robert írása röviden bemutatja a rendszermag-memóriafoglalás lépéseit, valamint azt is megtudhatjuk, hogy milyen változások történtek a 2.6-os rendszermagban.

A rendszermagfejlesztők bánatára a rendszermagban egyáltalán nem olyan egyszerű memóriát foglalni, mint a felhasználói térben. A nehézségek több okra vezethetők vissza, többek közt:

- A rendszermag körülbelül 1 GB virtuális és fizikai memóriakorláttal bír.
- A rendszermemória nem lapozható.
- A rendszermag fizikailag általában folytonos memóriát igényel.
- A magnak gyakran pihenés nélkül kell memóriát foglalnia.
- A rendszermagban elkövetett hibákért sokkal magasabb árat fizetünk, mint bárhol másutt.

Bár a nagy memóriatömegek elérése nyilvánvalóan nem kifejezetten a rendszermagnak kijáró fényűzés, a nehézségek tisztázásával sokat tehetünk azért, hogy a folyamat viszonylag fájdalommentes legyen.

Általános célú foglaló (allocator)

A memóriafoglalás általános célú felülete a rendszermagban a `kmalloc()`:

```
#include <linux/slab.h>
void * kmalloc(size_t size, int flags);
```

Ismerősnek tűnhet – végtére is éppen olyan, mint a felhasználói térben megszokott `malloc()` – kivéve, hogy egy második, `flags` nevű kapcsolót is vár. Most egyelőre tekintsünk el tőle és lássuk, mit ismerünk! Először is a méret (size) itt is ugyanazt jelenti, mint a `malloc()`-változatban: a foglalás méretét adja meg bájtban. Sikeres visszatérés esetén a `kmalloc()` a memória `size` méretű területére ad vissza mutatót. A legfoglalt memória kiosztása bármilyen objektumtípus tárolásához és eléréséhez megfelelő. Akárcsak a `malloc()`, a `kmalloc()` is lehet sikertelen, ezért ellenőriznünk kell, hogy a visszatérési érték nem `NULL`-e. Lássunk egy példát!

```
struct falcon *p;

p = kmalloc(sizeof (struct falcon), GFP_KERNEL);
if (!p)
  /* a foglalás sikertelen - itt kezeljük le*/
```

Zászlók (Flags)

A `flags` mező a memóriafoglalás viselkedését befolyásolja. A jelzőbiteket három csoportra bonthatjuk: műveletmódosítók, zónamódosítók és típusok. A műveletmódosító mondja meg a rendszermagnak, hogy miképp foglalja le a memóriát. Például ezek mutatják meg, hogy a rendszermag pihenhet-e (vagyis a `kmalloc()` hívás blokkolhat-e) a foglalás teljesítése közben. Ezzel szemben a zónamódosítók arról tájékoztatják a rendszermagot, hogy honnan kell a kérést kielégíteni. Például bizonyos kérések teljesítésekor olyan memóriát kell felhasználni, amit bármelyik eszköz a közvetlen memóriaelérés (Direct Memory Access, DMA) szolgáltatáson keresztül közvetlenül el tud érni. Végül a típuszászlók mutatják a foglalás típusát; ezek a megfelelő zóna- és

1. táblázat **A műveletmódosítók**

Zászló	Leírás
__GFP_COLD	A rendszermag a hideg gyorstárlapokat (cache cold pages) használja.
__GFP_FS	A rendszermag fájlrendszer ki–bemenetet indíthat el.
__GFP_HIGH	A rendszermag hozzáférhet a vésztartalékokhoz (emergency pools).
__GFP_IO	A rendszermag lemez-B-K-műveleteket végezhet.
__GFP_NOFAIL	A rendszermag megismételheti a foglalást.
__GFP_NORETRY	A rendszermag nem tér vissza, ha a foglalás nem sikerült.
__GFP_NOWARN	A rendszermag sikertelenség esetén nem ír figyelmeztetéseket.
__GFP_REPEAT	A rendszermag megismétli a foglalást, ha nem jár sikerrel.
__GFP_WAIT	A rendszermag alhat.

2. táblázat **A zónamódosítók**

Zászló	Leírás
__GFP_DMA	Kizárólag DMA-k által elérhető memória foglalása.
No flag	Onnan foglalunk, ahol van.

műveletmódosítókat foglalják össze egyetlen szimbólumban. Általában több zóna és művelet megadása helyett egyszerűen csak egy típust adunk meg.
Az *1. táblázat* a műveletmódosítókat sorolja fel, míg a *2. táblázatban* a zónamódosítókat találjuk. Több eltérő jelet is használhatunk; a rendszermag memóriafoglalása nem egyértelmű. A rendszermagban a memóriafoglalás számos jellemzőjét befolyásolhatjuk. Akkor járunk el helyesen, ha kódunk a típusjeleket és nem az egyedi művelet- és zónamódosítókat használja. A leggyakoribb zászlók: a `GFP_ATOMIC` és a `GFP_KERNEL`; szinte valamennyi rendszermag-memóriafoglalás e két zászló közül használja valamelyiket.
A `GFP_ATOMIC` jel arra utasítja a memóriafoglalót, hogy soha ne blokkoljon. Ezt a zászlót olyan helyzetekben használjuk, amikor nem szabad „aludni" – azaz atomi jellegűnek kell maradni –, például a megszakításkezelőkben, az alsó részben (bottom half) és az olyan folyamat jellegű kódokban, amelyek zárakat tartalmaznak. Minthogy a rendszermag nem blokkolhatja a foglalást és megpróbálja felszabadítani a kérelem kielégítéséhez szükséges memóriát, a `GFP_ATOMIC` zászlót megadó kérelemnek kisebb esélye van a sikerre, mint azoknak, amelyek nem igénylik ezt a zászlót. Ugyanakkor megoldásunk az alvással nem egyeztethető össze, tehát ez az egyetlen lehetőség. A `GFP_ATOMIC` használata nagyon egyszerű:

```
struct wolf *p;

p = kmalloc(sizeof (struct wolf), GFP_ATOMIC);
if (!p)
    /* hiba */
```

Ennek megfelelően a `GFP_KERNEL` zászló a normál rendszermagfoglalást jelenti. Ezt a zászlót használjuk az olyan folyamatokban, amelyekben nincsenek zárak. Az ilyen zászlóval meghívott `kmalloc()` alhat; ezért ezt a zászlót csak akkor szabad használnunk, ha nem okoz galibát.

3. táblázat **A típusok**

Zászló	Leírás
GFP_ATOMIC	A foglalás nagy fontosságú és nem alszik. Ezt a jelet használjuk a megszakításkezelőkben, az alsó részben (bottom half) és egyéb helyzetekben, ahol az alvás nincs megengedve.
GFP_DMA	A DMA-k által elérhető memória foglalása. Azok az eszközmeghajtók, amelyek DMA-elérhetőségű memóriát alkalmaznak, ezt a zászlót használják.
GFP_KERNEL	Normál, blokkolható foglalás. Folyamatkörnyezetben ezt a zászlót használjuk, amennyiben az alvás biztonságos.
GFP_NOFS	Ez a fajta foglalás blokkolható és lemezműveletek kezdeményezését is megengedi, de a fájlrendszerműveleteket nem. Ezt a jelet használjuk a fájlrendszerkódokban, ahol másik fájlrendszerművelet indítása nem kívánatos.
GFP_NOIO	Ez a foglalás blokkolható, de nem kezdeményezhet blokkos ki- és bemenetet. Ezt a jelet használjuk a blokkos rétegben, amelyben nem szeretnénk további blokkos B-K-műveleteket indítani.
GFP_USER	Normál, blokkolható foglalás. Ezt a zászlót használjuk a felhasználói folyamatok memóriafoglalásához.

A rendszermag szükség esetén az alvás képességét a memória felszabadítására használja, ezért aztán az ilyen zászlót használó foglalásoknak nagyobb az esélyük a sikerre. Amennyiben például nem áll elegendő memória rendelkezésre, a rendszermag blokkolhatja a kódot és a lemezre menthet néhány nem működő lapot, csökkentheti a memóriagyorstár méretét, kiírhatja a kimeneti átmeneti tárat (buffer) és így tovább.
Néha, például amikor ISA eszközvezérlőt készítünk, biztosítanunk kell, hogy a lefoglalt memória képes együttműködni a kiszolgáló DMA-rendszerrel. Az ISA eszközök esetében ez a fizikai memória első 16 MB-nyi területét jelenti. Ha bizonyosak akarunk lenni benne, hogy a rendszermag ebből a területből foglal memóriát, a `GFP_DMA` zászlót kell használnunk. Ezt általában a `GFP_ATOMIC` vagy a `GFP_KERNEL` zászlóval együtt használjuk; a jeleket bináris `OR` művelettel kapcsolhatjuk össze. Például, ha a rendszermagot DMA-ra alkalmas memóriaterület foglalására akarjuk utasítani és engedélyezzük az alvást, az utasítás a következő módon fest:

```
char *buf;

/* DMA-ra alkalmas memóriát szeretnénk és szükség
   ➥esetén alhat*/
buf = kmalloc(BUF_LEN, GFP_DMA | GFP_KERNEL);
if (!buf)
    /* hiba */
```

A *3. táblázat* a zászlótípusokat sorolja fel, míg a *4. táblázatban* azt mutatjuk be, hogy az egyes zászlók milyen művelet- és zónamódosítóknak felelnek meg. A `<linux/gfp.h>` fejléc azonosítja a zászlókat.

Memória felszabadítása

Miután a `kmalloc()` segítségével lefoglalt memória használatát befejeztük, vissza kell adnunk a rendszermagnak a lefoglalt területet. Ezt a feladatot a `kfree()` végzi el, amely a felhasználói tér `free()` könyvtárhívásának felel meg.
A `kfree()` meghatározása így fest:

```
#include <linux/slab.h>

void kfree(const void *objp);
```

A `kfree()` használata azonos felhasználó térbeli megfelelőjével. Tegyük fel, hogy `p` a `kmalloc()` által lefoglalt

4. táblázat **A típusjelek összehasonlítása**

Zászló	Érték
GFP_ATOMIC	__GFP_HIGH
GFP_NOIO	__GFP_WAIT
GFP_NOFS	(__GFP_WAIT | __GFP_IO)
GFP_KERNEL	(__GFP_WAIT | __GFP_IO | __GFP_FS)
GFP_USER	(__GFP_WAIT | __GFP_IO | __GFP_FS)
GFP_DMA	__GFP_DMA

memóriaterület mutatója. Ebben az esetben a következő parancs szabadítaná fel a memóriablokkot:

```
kfree(p);
```

Akárcsak a felhasználói tér `free()` függvénye esetében, ha a `kfree()`-t egy már korábban felszabadított memóriaterülettel hívjuk meg vagy olyan mutatót adunk át, amely nem a `kmalloc()` függvénytől származik, hibázunk, ez pedig memóriahibákhoz vezethet. Fontos, hogy kiegyensúlyozottan hívjuk a foglalásokat és a felszabadításokat, ügyelve arra, hogy a `kfree()`-t mindig pontosan egyszer hívjuk meg a megfelelő mutatóhoz. A `kfree()` meghívása a `NULL` értékkel biztonságos, ugyanis az érték létezését ellenőrzi – mindazonáltal nem éppen értelmes ötlet.
Vessünk egy pillantást a teljes foglalásfelszabadítás ciklusra:

```
struct sausage *s;

s = kmalloc(sizeof (struct sausage), GFP_KERNEL);
if (!s)
    return -ENOMEM;
/* ... */

kfree(s);
```

Virtuális memória foglalása

A `kmalloc()` függvény fizikai és következésképpen virtuálisan folytonos memóriaterületet ad vissza. Ebben különbözik a felhasználói tér `malloc()`s függvényétől, amely szintén virtuális, de nem szükségszerűen folytonos fizikai memóriaterületet foglal le. A fizikailag folytonos területek két elsődleges előnnyel bírnak: először is sok alkatrész nem képes virtuális memóriát címezni. Ezért aztán, hogy egy memóriaterületet elérhessenek, a memóriablokknak fizikailag folytonos memóriaterületen kell léteznie. Másodszor a fizikailag folytonos memóriablokk egyetlen nagy lapterületet használ. Ez nagymértékben csökkenti a TLB (translation lookaside buffer) memóriacímző munkáját, hiszen mindössze egyetlen TLB-bejegyzésre van szükség.
A fizikailag folyamatos memóriának akad egy nagy hátránya: gyakran igen nehéz fizikailag folytonos memóriaterületet találni, s különösen igaz ez a nagyméretű foglalásokra. A csak virtuálisan folytonos memória sikeres foglalására jóval nagyobb esélyünk van. Amennyiben nincs szükségünk fizikailag folytonos memóriára, használjuk a `vmalloc()` függvényt:

```
#include <linux/vmalloc.h>

void * vmalloc(unsigned long size);
```

A `vmalloc()` által lefoglalt memóriát a `vfree()` segítségével adhatjuk vissza a rendszernek:

```
#include <linux/vmalloc.h>

void vfree(void *addr);
```

A `vfree()` használata ismét azonos a felhasználói tér `malloc()` és `free()` függvényének a használatával:

```
struct black_bear *p;

p = vmalloc(sizeof (struct black_bear));
if (!p)
    /* hiba */

/* ... */

vfree(p);
```

Ebben az esetben a `vmalloc()` alhat.
A rendszermagban nagyszámú foglalás használhat `vmalloc()` függvényt, mivel csak kevés foglalásnak kell egybefüggőnek látszania az eszközök számára. Ha olyan memóriát foglalunk le, amit csak programok érnek el, például felhasználói folyamatokkal kapcsolatos adatokat tárolunk, a memóriának nem szükséges fizikailag folytonosnak lennie. Ennek ellenére a rendszermagban kevés foglalás használja a `vmalloc()` függvényt. A legtöbb a `kmalloc()`-ot választja, mégha szükségtelen is, részben történelmi, részben teljesítménynövelési okokból kifolyólag. Minthogy a fizikailag folytonos lapok jóval kevesebb munkát adnak a TLB-nek, az ebből fakadó teljesítménynövekedést gyakran sokra tartják. Ennek ellenére, ha megabájtok tucatjait akarjuk lefoglalni a rendszermagban, a `vmalloc()` a legjobb választás.

Apró, állandó méretű verem

A felhasználói folyamatokkal ellentétben a rendszermagban futó kódnak nincsen nagy, dinamikusan növekvő verme – ehelyett a rendszermag minden folyamata saját apró, állandó méretű veremmel rendelkezik. A verem pontos mérete kiépítésfüggő; a legtöbb architektúra két lapot foglal le veremnek, a verem tehát a 32 bites gépeken 8 KB.
Az apró verem miatt kerülendők a nagy, önműködő, veremben tárolt foglalások. Ha megnézzük, soha sem találunk ilyesmit a rendszermagkódban:

```
#define BUF_LEN        2048

void rabbit_function(void)
{
    char buf[BUF_LEN];
    /* ...  */
}
```

A helyes változat a következő:

```
#define BUF_LEN  2048

void rabbit_function(void)
{
    char *buf;

    buf = kmalloc(BUF_LEN, GFP_KERNEL);
    if (!buf)
        /* error! */

/* ... */
}
```

Felhasználói térben ritkán látunk hasonlót, hiszen nincs túl sok értelme dinamikusan foglalni memóriát, ha a kód írásakor már ismerjük a lefoglalandó méretet. A rendszermagban azonban mindig dinamikus foglalást alkalmazunk, amennyiben a foglalás néhány bájtnál nagyobb. Így elkerülhetjük a veremtúlcsordulást, ami egyébként számos gondot okozhatna.

Összegzés

Fellebbentettük a fátylat a rendszermag memóriakezeléséről, és láthattuk, hogy nem sokkal nehezebb, mint a felhasználói térben található megfelelője. Néhány ökölszabály, amit érdemes betartanunk:

- Döntsük el, hogy alhatunk-e (azaz a `kmalloc()` blokkolható-e) vagy sem. Ha egy megszakításkezelő közepén vagyunk és az alsó részben (bottom half) egy zárat tartunk, akkor ezt nem tehetjük meg. Amennyiben folyamatszinten vagyunk és nem tartunk zárakat, valószínűleg megtehetjük.
- Ha alhatunk, a `GFP_KERNEL`-t adjuk meg.
- Ha nem, válasszuk a `GFP_ATOMIC` zászlót.
- Amennyiben DMA által használható memóriára van szükségünk (például ISA vagy hibás PCI eszközhöz), adjuk meg a `GFP_DMA` zászlót.
- Mindig ellenőrizzük és kezeljük a `kmalloc()` által esetleg visszaadott `NULL` értéket.
- Ne folyassuk el a memóriát; ne felejtsük el a `kfree()` függvényt valahol meghívni.
- Biztosítsuk, hogy a `kfree()`-t nem hívjuk meg többször, és sehol se próbáljuk meg a már felszabadított memóriarészeket elérni.

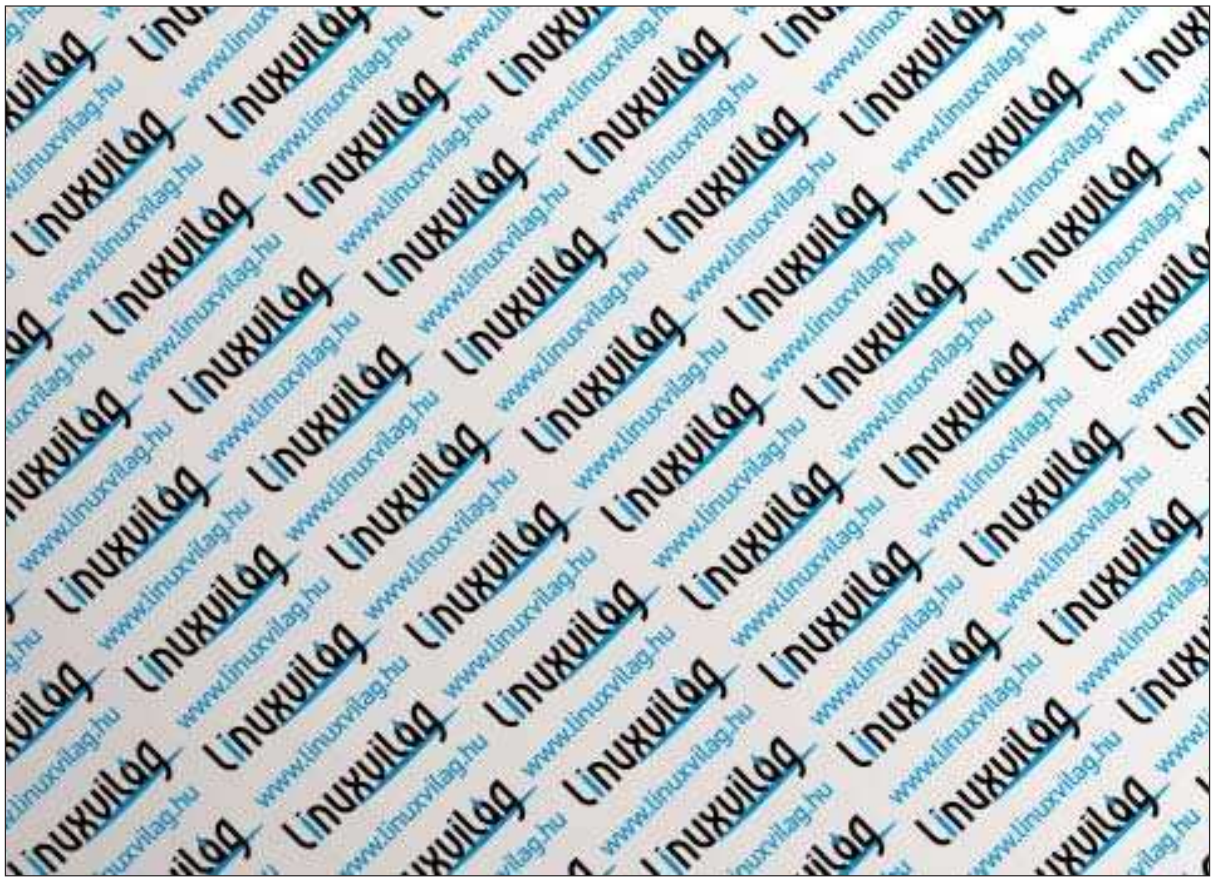
Linux Journal 2003. december, 116. szám

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

TOVÁBBI ÉRDEKESSÉGEK

További adatokat a rendszermagfa fájljaiban találunk:

- include/linux/gfp.h: a foglalási zászlók gyűjteménye.
- include/linux/slab.h: a `kmalloc()` és társainak meghatározásai.
- mm/page_alloc.c: lapfoglalási függvények.
- mm/slab.c: a `kmalloc()` és társainak megvalósítása.



I2C illesztőprogramok (1. rész)

Az I2C-busz segítségével figyelemmel követhetjük számítógépünk egészségi állapotát. Tekintsük át, hogyan írhatunk egy minden szükséges állapotadatot begyűjtő illesztőprogramot!

A Linuxvilág 2003. júniusi és augusztusi számában a Linux-rendszermag illesztőprogram-modelljével foglalkoztam, példaként pedig az I2C alrendszert hoztam fel. Írásomban az I2C alrendszer feladataival és a hozzá való illesztőprogramok írásával fogok foglalkozni.
Az I2C egy, két vezetéket használó soros buszprotokoll neve, ami eredetileg a Philips fejlesztése. Beágyazott rendszerekben gyakran használják a különféle összetevők közötti kapcsolattartásra, a PC-s alaplapok pedig az érzékelőlapkákkal való párbeszéd céljából alkalmazzák. Az érzékelők általában a ventilátorok fordulatszámát, a processzor hőmérsékletét és más alkatrészek egyéb jellemzőit mérik és jelentik. Bizonyos RAM-lapkák is ilyen protokollon keresztül továbbítják a DIMM-modullal kapcsolatos adatokat az operációs rendszernek.
Az I2C rendszermagkód életének jelentős részét a fő rendszermagfából kimaradva töltötte, fejlesztése egyébként a 2.0-s rendszermag idején kezdődött. A 2.4-es rendszermag már tartalmaz valamennyi I2C-támogatást, elsősorban bizonyos megjelenítő-illesztőprogramokhoz. A 2.6-os rendszermaggal az I2C-kód jelentős része bekerült a fő rendszermagfába, azoknak rendszermag-fejlesztőknek köszönhetően, akik az adatcsere-felületeket a rendszermag-fejlesztői közösség számára elfogadhatóbbakká alakították át. Néhány illesztőprogram még most is csak a külső CVS-fában található meg, a fő ➲ http://kernel.org fában nem, de csupán idő kérdése, hogy ezeknek az átültetése is megtörténjen.
Az I2C rendszermagkód logikailag több részre bontható, amelyek a következők: I2C-mag, I2C buszillesztőprogramok, I2C algoritmus-illesztőprogramok és I2C lapka-illesztőprogramok. Írásomban az I2C-mag működésével nem foglalkozom, inkább a busz- és az algoritmus-illesztőprogramok írására összpontosítok. Az I2C lapka-illesztőprogramok írásának témájára a második részben kerül sor.

I2C buszillesztőprogramok

Az I2C buszillesztőprogramok leírása az `i2c_adapter` adatszerkezetben található meg, ennek megadása az ***include/linux/i2c.h*** fájlban szerepel. A buszillesztőprogramnak csak a következő mezőknek kell értéket adnia:

- `struct module *owner` – értéke (`THIS_MODULE`) segítségével a modulra való hivatkozások számlálása oldható meg.
- `unsigned int class` – az illesztőprogram által támogatott I2C osztályeszközök. Értéke általában I2C_ADAP_CLASS_SMBUS.
- `struct i2c_algorithm *algo` – mutató az i2c_algorithm adatszerkezetre, amely az I2C buszvezérlőn folyó adatcsere mikéntjét írja le. Az adatszerkezetről bővebb leírást is fogok adni.
- `char name[I2C_NAME_SIZE]` – az I2C buszillesztőprogram beszédes neve. Az itt megadott érték az I2C-csatolóhoz tartozó `sysfs` fájlnévben jelenik meg.

Az alábbi kódrészlet – amely az `i2c_adapter` adatszerkezet értékadását szemlélteti – egy ***tiny_i2c_adap.c*** nevű I2C-csatoló példa-illesztőprogramból származik. Az illesztőprogram az 57. CD Magazin/I2C könyvtárában található.

```
static struct i2c_adapter tiny_adapter = {
    .owner   = THIS_MODULE,
    .class   = I2C_ADAP_CLASS_SMBUS,
    .algo    = &tiny_algorithm,
    .name    = "tiny adapter",
};
```

Az I2C-csatoló bejegyzését az illesztőprogram az `i2c_add_adapter` függvény meghívásával végzi el, amelynek egy, az `i2c_adapter` adatszerkezetre hivatkozó mutatót ad át:

```
retval = i2c_add_adapter(&tiny_adapter);
```

Ha az I2C-csatoló olyan típusú eszköz felett üzemel, amelyhez `device` adatszerkezet van hozzárendelve – ilyenek például a PCI- vagy USB-eszközök –, akkor az `i2c_add_adapter` meghívása előtt a csatolóeszköz szülőjének a mutatóját erre az eszközre kell állítani.
A ***drivers/i2c/busses/i2c-piix4.c*** illesztőprogramban a mutató beállítása például a következőképpen történik:

```
/* sysfs csatolás beállítása a szülőeszközre */
piix4_adapter.dev.parent = &dev->dev;
```

Ha a mutató értékét nem adjuk meg, akkor az I2C-csatoló

1. lista A tiny_access függvény

```
static s32 tiny_access(struct i2c_adapter *adap,
                       u16 addr,
                       unsigned short flags,
                       char read_write,
                       u8 command,
                       int size,
                       union i2c_smbus_data *data)
{
 int i, len;

 dev_info(&adap->dev, "%s hívására a következő
 ➥átadott értékekkel "
        "került sor:\n",
        __FUNCTION__);
 dev_info(&adap->dev, "cím = %.4x\n", addr);
 dev_info(&adap->dev, "kapcsolók = %.4x\n",
flags);
 dev_info(&adap->dev, "olvasas_iras = %s\n",
          read_write == I2C_SMBUS_WRITE ?
          "iras" : "olvasas");
 dev_info(&adap->dev, "parancs = %d\n",
          command);

  switch (size) {
 case I2C_SMBUS_PROC_CALL:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_PROC_CALL\n");
    break;
 case I2C_SMBUS_QUICK:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_QUICK\n");
    break;
 case I2C_SMBUS_BYTE:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_BYTE\n");
    break;
 case I2C_SMBUS_BYTE_DATA:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_BYTE_DATA\n");
    if (read_write == I2C_SMBUS_WRITE)
        dev_info(&adap->dev,
                 "adat = %.2x\n", data->byte);
    break;
 case I2C_SMBUS_WORD_DATA:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_WORD_DATA\n");
    if (read_write == I2C_SMBUS_WRITE)
        dev_info(&adap->dev,
                 "adat = %.4x\n", data->word);
    break;
 case I2C_SMBUS_BLOCK_DATA:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_BLOCK_DATA\n");
    if (read_write == I2C_SMBUS_WRITE) {
        dev_info(&adap->dev, "adat = %.4x\n",
                 data->word);
        len = data->block[0];
        if (len < 0)
            len = 0;
        if (len > 32)
            len = 32;
        for (i = 1; i <= len; i++)
            dev_info(&adap->dev,
                     "adat->blokk[%d] = %x\n",
                     i, data->block[i]);
    }
    break;
 }
 return 0;
}
```

a `legacy` (örökölt) buszra kerül, és a `sysfs` fában a */sys/devices/legacy* elérési út alatt látszik. Nézzük, hogy mi történik példa-illesztőprogramunkkal bejegyzéskor:

```
$ tree /sys/devices/legacy/
/sys/devices/legacy/
|-- detach_state
|-- floppy0
|   |-- detach_state
|   `-- power
|       `-- state
|-- i2c-0
|   |-- detach_state
|   |-- name
|   `-- power
|       `-- state
`-- power
    `-- state
```

Mint a rendszermag illesztőprogram-modelljével foglalkozó korábbi írásokból kiderül, az I2C-csatoló a */sys/class/i2c-adapter* könyvtárban is megjelenik:

```
$ tree /sys/class/i2c-adapter/
/sys/class/i2c-adapter/
`-- i2c-0
    |-- device -> ../../../devices/legacy/i2c-0
    `-- driver ->
../../../bus/i2c/drivers/i2c_adapter
```

Az I2C-csatoló bejegyzésének törlését az illesztőprogram az `i2c_del_adapter` függvény – átadott értéke egy, az `i2c_adapter` adatszerkezetre hivatkozó mutató – meghívásával végezheti el:

```
i2c_del_adapter(&tiny_adapter);
```

Az I2C algoritmus-illesztőprogramok

Az I2C-algoritmust az I2C buszillesztőprogram használja az I2C-busszal való kapcsolattartásra. A legtöbb I2C buszillesztőprogram saját I2C-algoritmust ad meg és használ. Az algoritmus jellege szorosan összefügg a buszillesztőprogram

és az adott eszköz közti párbeszéd módjával. Bizonyos I2C buszillesztőprogram-osztályokhoz már több I2C algoritmus-illesztőprogram is készült; példaként a ***drivers/i2c/i2c-algo-ite.c*** fájl által kezelt ITE csatolókat, a ***drivers/i2c/i2c-algo-ibm_ocp.c által*** vezérelt IBM PPC 405 csatolókat és a ***drivers/i2c/i2c-algo-bit.c*** fájlban található általános I2C bit-eltoló algoritmust említhetném. Mindegyik algoritmus saját függvényekkel bír, használatukhoz az I2C buszillesztőprogram oldaláról külön bejegyzés szükséges. Minderről a rendszermagfa ***drivers/i2c/i2c-algo-*.c*** fájljaiban találhatunk bővebb tudnivalókat.

Példa-illesztőprogramunkhoz saját I2C algoritmus-illesztőprogramot fogunk készíteni. Az algoritmus-illesztőprogram megadása az `i2c_algorithm` adatszerkezetben, az ***include/linux/i2c.h*** fájlban található. A gyakran használt mezők közül íme néhánynak a leírása:

- `char name[32];` – az algoritmus neve.
- `unsigned int id;` – az adatszerkezet által megadott algoritmus típusának a leírása. A különféle típusok megadása az ***include/linux/i2c-id.h*** fájlban található. Az összes típus neve az `I2C_ALGO_` karakterlánccal kezdődik.
- `int (*master_xfer)(struct i2c_adapter *adap,struct i2c_msg msgs[], int num);` – függvénymutató, értéket akkor szükséges neki adni, ha az adott algoritmus-illesztőprogram közvetlen I2C-elérést is végezhet. Ha az értékét megadjuk, akkor mindig ennek a függvénynek a meghívására kerül sor, ha egy I2C lapka-illesztőprogram kapcsolatba szeretne lépni a lapkával. Ha az értéke `NULL`, akkor ezt a szerepet az `smbus_xfer` függvény veszi át.
- `int (*smbus_xfer) (struct i2c_adapter *adap, u16 addr, unsigned short flags, char read_write, u8 command, int size, union i2c_smbus_data *data);` – függvénymutató, akkor kell neki értéket adni, amikor az algoritmus-illesztőprogram hozzáférhet az SMB-buszhoz. A legtöbb PCI alapú I2C-buszillesztőprogram képes erre, tehát esetükben ennek a függvénymutatónak értéket kell adnunk. Ha ezt megtesszük, akkor mindig ennek a függvénynek a meghívására kerül sor, amennyiben egy I2C lapka-illesztőprogram kapcsolatba szeretne lépni a lapkával. Ha az értéke `NULL`, akkor ezt a szerepet a `master_xfer` függvény veszi át.
- `u32 (*functionality) (struct i2c_adapter *);` – függvénymutató, a megadott függvényt az I2C-mag hívja meg annak meghatározásához, hogy az I2C-csatoló illesztőprogram milyen jellegű írásokra és olvasásokra képes.

Az I2C-csatoló példa-illesztőprogramunkban az `i2c_adapter` adatszerkezet a `tiny_algorithm` változóra hivatkozott. Ennek az adatszerkezetnek a megadása a következő:

```
static struct i2c_algorithm tiny_algorithm = {
    .name           = "tiny algorithm",
    .id             = I2C_ALGO_SMBUS,
    .smbus_xfer     = tiny_access,
    .functionality  = tiny_func,
};
```

A `tiny_func` függvény egészen kisméretű, a feladata mindössze az, hogy az I2C-magot tájékoztassa az algoritmus által támogatott I2C üzenettípusokról. Ettől az illesztőprogramtól az alábbi néhány I2C üzenettípus támogatását várjuk el:

```
static u32 tiny_func(struct i2c_adapter *adapter)
{
    return I2C_FUNC_SMBUS_QUICK |
           I2C_FUNC_SMBUS_BYTE |
           I2C_FUNC_SMBUS_BYTE_DATA |
           I2C_FUNC_SMBUS_WORD_DATA |
           I2C_FUNC_SMBUS_BLOCK_DATA;
}
```

Az összes I2C üzenettípus az ***include/linux/i2c.h*** fájlban van megadva, és az `I2C_FUNC_` karakterlánccal kezdődik.

A `tiny_access` függvény meghívására akkor kerül sor, amikor egy I2C ügyfél-illesztőprogram az I2C-busszal szeretne párbeszédet kezdeményezni. Példafüggvényünk meglehetősen egyszerű: mindössze naplózza az I2C lapka-illesztőprogram kéréseit a `syslog`-ba, valamint a hívó felé jelenti a sikeres végrehajtást. A napló alapján minden különböző cím- és adattípus összesíthető, amelyet egy I2C lapka-illesztőprogram valaha is igényelhet. A példafüggvény megvalósítása az *1. listán* látható.

2. lista Az „lm75” I2C ügyfél-illesztőprogram

```
$ modprobe lm75
$ tree /sys/bus/i2c/
/sys/bus/i2c/
|-- devices
|   |-- 0-0048 -> ../../../devices/legacy/i2c-0/0-0048
|   |-- 0-0049 -> ../../../devices/legacy/i2c-0/0-0049
|   |-- 0-004a -> ../../../devices/legacy/i2c-0/0-004a
|   |-- 0-004b -> ../../../devices/legacy/i2c-0/0-004b
|   |-- 0-004c -> ../../../devices/legacy/i2c-0/0-004c
|   |-- 0-004d -> ../../../devices/legacy/i2c-0/0-004d
|   |-- 0-004e -> ../../../devices/legacy/i2c-0/0-004e
|   `-- 0-004f -> ../../../devices/legacy/i2c-0/0-004f
`-- drivers
    |-- i2c_adapter
    `-- lm75
        |-- 0-0048 -> ../../../../devices/legacy/i2c-0/0-0048
        |-- 0-0049 -> ../../../../devices/legacy/i2c-0/0-0049
        |-- 0-004a -> ../../../../devices/legacy/i2c-0/0-004a
        |-- 0-004b -> ../../../../devices/legacy/i2c-0/0-004b
        |-- 0-004c -> ../../../../devices/legacy/i2c-0/0-004c
        |-- 0-004d -> ../../../../devices/legacy/i2c-0/0-004d
        |-- 0-004e -> ../../../../devices/legacy/i2c-0/0-004e
        `-- 0-004f -> ../../../../devices/legacy/i2c-0/0-004f
```

Elkészült a tiny_i2c_adap illesztőprogram, lefordítottuk, betöltöttük – de mire jó? A válasz prózai lesz: önmagában semmire. Az I2C buszillesztőprogramnak I2C ügyfél-illesztőprogramra van szüksége ahhoz, hogy bármi értelmeset tudjon tenni, azon kívül, hogy elücsörög a sysfs fában.
Az lm75 I2C ügyfél-illesztőprogram betöltéskor tehát a tiny_i2c_adap illesztőprogram segítségével próbálja megtalálni azt a lapkát, amelyhez őt írták (lásd a 2. *listát*).
Mivel a tiny_i2c_adap illesztőprogram minden olvasási és írási kérésre sikert jelezve válaszol, az A tiny_access függvény lm75 I2C lapka-illesztőprogram azt hiszi, hogy az lm75 lapka lehetséges elérési címeinek mindegyikén talált egy ilyen lapkát. A bőséges címválaszték az oka annak, hogy a 0-0048 – 0-004f tartományban több I2C-eszköz is létrejött.
Ha benézünk valamelyik eszköz könyvtárába, akkor a lapka-illesztőprogram érzékelőfájljait láthatjuk:

```
$ tree /sys/devices/legacy/i2c-0/0-0048/
/sys/devices/legacy/i2c-0/0-0048/
|-- detach_state
|-- name
|-- power
|   `-- state
|-- temp_input
|-- temp_max
`-- temp_min
```

A ***detach_state*** fájlt és a ***power*** könyvtárat a rendszermag-illesztőprogram magja hozza létre, mindkettő az energiakezelésben jut szerephez. Ismétlem, ezeket nem az lm75 illesztőprogram hívja életre. Tekintsük át a könyvtár további állományainak a feladatát!
Ha az lm75 illesztőprogramból lekérdezzük a temp_max pillanatnyi értékét, a következőt kapjuk:

```
$ cat /sys/devices/legacy/i2c-0/0-0048/temp_max
1000
```

Az érték lekérdezéséhez az lm75 illesztőprogram az I2C-busz bizonyos címeiről végzett olvasásra kérte a tiny_i2c_adap illesztőprogramot. A kérést a syslog alapján lehet követni (lásd a 3. listát az 57. CD Magazin/IRC könyvtárában).
A napló alapján kiderül, hogy a tiny_access függvény meghívása három alkalommal történt meg. Az első parancs egy word típusú adatot akart kiolvasni a 0048-as című eszköz 0 számú regiszteréből. A második és a harmadik olvasás ugyanezen eszköz 3-as és 2-es regiszterére irányult.
A parancsok a ***drivers/i2c/chips/lm75.c*** fájlban található lm75_update_client függvény kódjának a következő részeivel társíthatók:

```
data->temp_input = lm75_read_value(client,
                        LM75_REG_TEMP);
data->temp_max = lm75_read_value(client,
                         LM75_REG_TEMP_OS);
data->temp_hyst = lm75_read_value(client,
                       LM75_REG_TEMP_HYST);
```

Ugyanezen fájl lm75_read_valu*e függvénye az alábbi kódot tartalmazza:

```
/* Minden regiszter word méretű, kivéve a
beállítóregisztert. Az LM75 a felsőbb helyi értékű
bájtot helyezi előre, pontosan fordítva ahhoz
képest, ahogy azt megszoktuk. */
static int lm75_read_value(struct i2c_client
                          *client, u8 reg)
{
  if (reg == LM75_REG_CONF)
       return i2c_smbus_read_byte_data(client,
                                reg);
  else
       return swap_bytes(
             i2c_smbus_read_word_data(client,
                                      reg));
}
```

Amikor tehát az lm75 illesztőprogram ki akarja olvasni a legnagyobb hőmérsékleti értéket, a regiszter számával meghívja az lm75_read_value függvényt, amely viszont az I2C-mag i2c_smbus_read_word_data függvényének adja át a szót. Az I2C-magnak ez a függvénye megkeresi, hogy az ügyféleszköz melyik I2C-buszon található, majd az átvitel végrehajtása érdekében az ehhez az I2C-buszhoz társított algoritmust hívja meg. A tiny_i2c_adap illesztőprogramunk tehát ezzel a módszerrel kapja meg az átvitel elvégzésére vonatkozó kérést.
Ha ugyanebbe a ***sysfs*** fájlba írni szeretnénk, az lm75 illesztőprogram ugyanilyen eljárással kéri meg a tiny_i2c_adap illesztőprogramot az I2C-busz megadott címére végzendő adatírásra. A syslog-ban a kérés adatai is megjelennek:

```
$ echo 300 > /sys/devices/legacy/i2c-0/
➥0-0048/temp_max
$ dmesg
i2c_adapter i2c-0: tiny_access hívására
➥a következő átadott értékekkel került sor:
i2c_adapter i2c-0: cím = 0048
i2c_adapter i2c-0: kapcsolók = 0000
i2c_adapter i2c-0: olvasas_iras = iras
i2c_adapter i2c-0: parancs = 3
i2c_adapter i2c-0: méret = I2C_SMBUS_WORD_DATA
i2c_adapter i2c-0: adat = 8000
```

Összegzés

Ez alkalommal összefoglaltuk az I2C illesztőprogram-alrendszer alapvető jellemzőit, valamint áttekintettük, hogyan lehet egyszerű I2C busz- és I2C algoritmus-illesztőprogramot írni, amely bármely meglévő I2C ügyfél-illesztőprogrammal együttműködik. A teljes illesztőprogram ***dmn-09-i2c-adap.c*** néven az 57. CD Magazin/I2C könyvtárában található.
Az I2C lapkaillesztőprogramok írásának témájával a második részben foglalkozunk.

Linux Journal 2003. december, 116. szám

Greg Kroah-Hartman (greg@kroah.com)
Jelenleg a Linux-rendszerek USB és gyors csatlakoztatású (PCI Hot Plug) egységei rendszermagba épített meghajtóprogramjainak a fejlesztője. Az IBM-nél dolgozik és rendszermaggal kapcsolatos kérdésekkel foglalkozik.

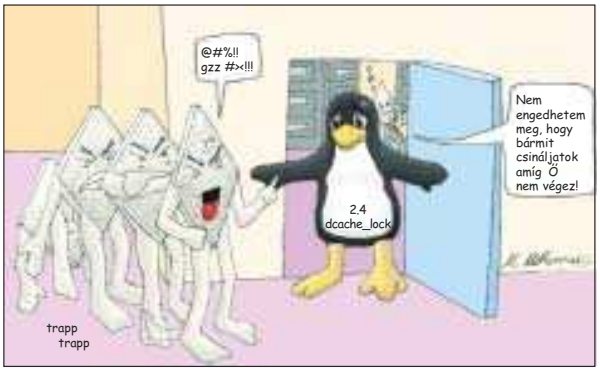
A dcache méretezése RCU-alapokon

A Linux fájlnévkeresési rendszerének újraszervezése sokat számíthat a komolyabb kiszolgálók méretezésekor.

Nemrég jelent meg 2.6-os Linux-rendszermagban az általában olvasott adatszerkezetekre kiélezett összehangolási módszer, az RCU (read-copy update, azaz olvasás–másolás–frissítés). Ebben a cikkben azt mutatjuk be, miképpen segíti az RCU a Linux könyvtárbejegyzés-gyorstárának (dcache) a méretezhetőségét. Ha a dolgok hátterére is kíváncsiak vagyunk, olvassuk el a „Using RCU in the Linux 2.5 Kernel” című írást a Linux Journal 2003. októberi számában.

Linux könyvtárbejegyzés-gyorstár

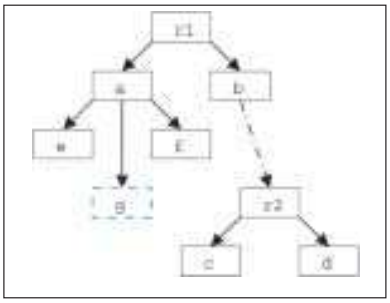
A Linux `dcache` a fájlrendszer hierarchiájának egy részét a memóriában kezeli. Ez a másolat költséges lemezműveletek nélkül is lehetővé teszi az útvonalkereséseket, amellyel nagymértékben növeli a fájlműveletek hatékonyságát.
A Linux-rendszermag a `mount` és `umount` műveletek megkönnyítése érdekében a befűzési fáról (mount tree) is tárol egy másolatot `struct vfsmount` szerkezetekben.
De ha a Linux 2.4 `dcache` ilyen nagyszerű, akkor miért kell megváltoztatni? A 2.4-es `dcache` nagy gondja, hogy globális `dcache_lock` zárat használ. Ez a zár kis rendszereken gyorstárvonalon fürgeséget, nagy rendszereken azonban a méretezhetőség komoly gátját jelenti, mint azt a *képen* is megfigyelhetjük.



Tux teszi a dolgát

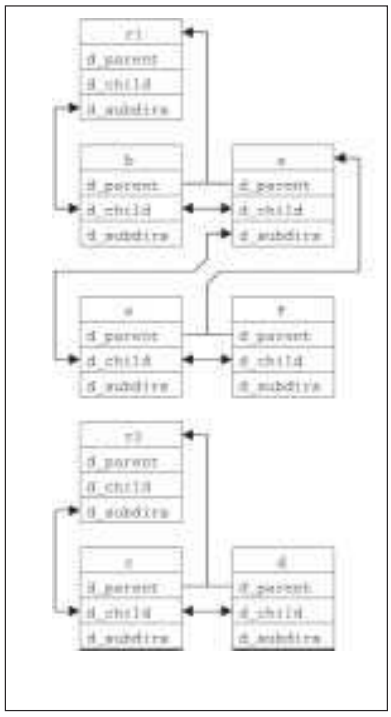
A dcache szemléletes bemutatása

Ebben a részben a cikkben később előforduló RCU-vonatkozású `dcache`-változásokkal ismerkedhetünk meg. A további részletekre kíváncsi olvasóinknak a forráskód átböngészését javasoljuk.
Írásunkban a *1. ábrán* bemutatott fájlrendszert fogjuk példaként felhasználni, amelyen két, `r1` és `r2` gyökérrel bíró befűzött fájlrendszert találunk. Mint azt a szaggatott nyíl mutatja, a második fájlrendszert a ***b*** könyvtár alá fűztük be. A ***g*** nevű állományt mostanában nem értük el, ezért nem is szerepel a `dcache`-ben – ezt szaggatott kék kerettel jelöltük.
A `dcache*` alrendszer a fájlrendszerfa több nézetét is kezelni tudja egyszerre. A *2. ábra* a könyvtárszerkezet megjelenítését mutatja be – minden egyes `dentry` egy-egy könyvtárnak felel meg, amely kétszeresen befűzött, `d_subdirs` mezővel kezdődő, körkörös listát tárol. Ez a lista fut végig a gyermek `dentry`-bejegyzések `d_child` mezőin. Minden gyermek `d_parent` mutatója értelemszerűen a szülőjére mutat. A befűzési pont (`dentry b`) nem közvetlenül a befűzött fájlrendszerre hivatkozik; ehelyett a befűzési hely `d_mounted` kapcsolóját állítjuk be, majd a `dcache` kikeresi a befűzött fájlrendszert a `mount_hashtable` táblázatából. Ezt a folyamatot a későbbiek folyamán ismertetjük.

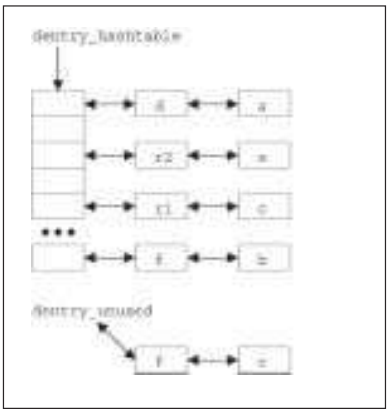


1. ábra Fájlrendszerfa-példa

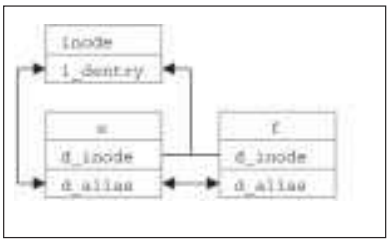
Bár a `d_subdirs` listákat közvetlenül is kereshetnénk, nagy könyvtárak esetében ez meglehetősen lassú megoldás lenne. Ezért az `__d_lookup()` inkább egy hasítótáblát készít a könyvtárak `dentry` mutatóihoz és a gyermekek nevéhez. Ezt a `dentry_hashtable`-t használja fel a megfelelő `dentry`-bejegyzéshez. A táblát a *3. ábra* mutatja be a `dentry_unused` elemmel kezdődő LRU listával együtt.
Az LRU lista minden `dentry` eleme általában a hasítótáblában (hash table) is benne van; kivételt képeznek azok az esetek, amelyekben a szülőkönyvtárak ideiglenesek, ilyen például az NFS és a hozzá hasonló osztott fájlrendszerek.
Minden `dentry` a `d_inode` mutató segítségével hivatkozik a hozzá tartozó fájlleíróra (inode). A `d_inode` mutató negatív `dentry`-bejegyzéseknél lehet `NULL` is, ami a hiányzó



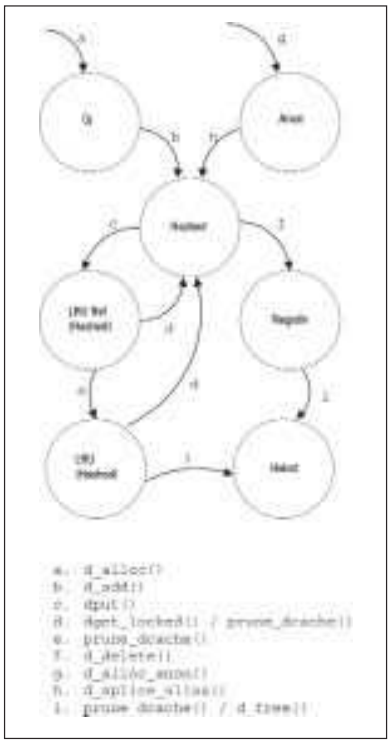
2. ábra Fájlrendszerfa-példánk dcache-megvalósítása



3. ábra dentry hasítótábla



4. ábra
Közvetlen befűzésű álnévláncok

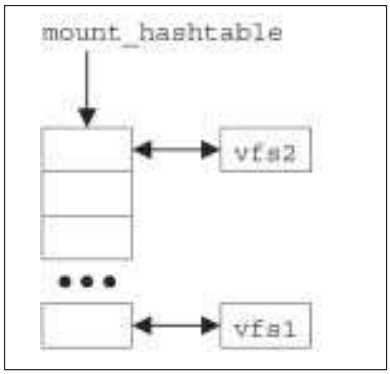


5. ábra A dentry állapotdiagram

fájlleíróra utal. Negatív `dentry` jöhet létre például akkor, amikor a fájlrendszer törli a `dentry` fájlját, vagy ha valaki zárolni próbál egy nemlétező állományt. A negatív `dentry` elemek felgyorsíthatják a rendszer teljesítményét, hiszen így a nemlétező fájlok ismételt elérési kísérlete során nem kell folyton az alsóbb szintű fájlrendszerhez fordulni. Ehhez hasonló módon a közvetlen befűzések is azonos `dentry`-bejegyzéseken osztozkodnak, mint azt az *4. ábrán* láthatjuk. A *5. ábrán* egy magas szintű `dentry` állapotdiagramot láthatunk. A normál `dentry` életciklus a következő:

1. A `d_alloc()` lefoglalja az új `dentry`-t az újonnan hivatkozott állományhoz, ezzel belépünk a ***New*** (új) állapotba.
2. A `d_add()` nevet és fájlleírót rendel az új bejegyzéshez, ezzel beléptünk a ***Hashed*** állapotba.
3. Miután végzett a fájllal, a `d_put()` felveszi a `dentry`-t a LRU listába és beállítja a `DCACHE_REFERENCED` bitet a `d_vfs_flags` mezőben – így jutunk el az ***LRU Ref (Hashed)*** állapotba.
4. Amennyiben az állományra az ***LRU Ref (Hashed)*** állapotban ismét hivatkoznak, a `dget_locked()` – amelyet általában a `d_lookup()` hív meg – használatra jelöli ki. Ha a következő `prune_dcache()` híváskor még mindig használatban van, eltávolítódik az LRU listáról és ismét visszakerül a ***Hashed*** állapotba.
5. Egyébként a `prune_dcache()` eltávolítja a `DCACHE_REFERENCED` bitet a `d_vfs_flags` mezőből, és ilyen módon eljutunk az ***LRU (Hashed)*** állapotba.
6. Akárcsak az előbbi esetben, ha a fájlra ismét hivatkoznak, a `dget_locked()` használatra jelöli be, hogy azután a `prune_dcache()` eltávolíthassa a LRU listáról és ismét ***Hashed*** állapotba kerüljön át.
7. Egyébként a második egymást követő `prune_dcache()` hívás után a `d_free()` függvényt hívjuk meg a `prune_one_dentry()` függvényen keresztül, és végül ***Dead (halott)*** állapotba kerül.

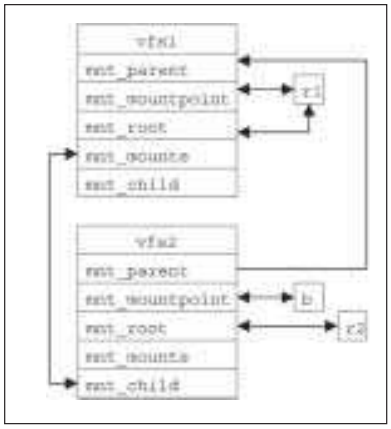
A *6. ábrából* kiolvasható egyéb útvonalak is elképzelhetőek. Például amikor egy osztott fájlrendszer gyorstárazott fájlmutatót alakít új `dentry`-bejegyzéssé, az `_alloc_anon()` függvény segítségével lefoglal egy `dentry`-t, miközben az objektum szülője e `dentry` gyorstárban már nem is létezik. Hasonlóképpen, amikor a `d_delete()` segítségével kitöröljük az adott `dentry` alatti állományt vagy könyvtárat, a `dentry`-t ***Negatív*** állapotba helyezzük. Ez a következő lépésben halott (***Dead***) állapotba jut. A 6. ábra a `mount_hashtable` adatszerkezetet mutatja be, amelyet a befűzési pont `dentry` és a befűzött fájlrendszer `struct vfsmount` szerkezetének összerendelésére használunk fel. Ez az összerendelés a mutatókat és a befűzési pont `dentry`-ket, valamint a mutatót és a befűzési pontot tartalmazó fájlrendszer `struct vfsmount` szerkezetét szervezi (pontosabban hasheli) össze. A `dentry` mutató és a `struct`



6. ábra Befűzési pontok átvitele

1. lista Zárnélküli útvonal–név–szakasz–keresés

```
1  struct dentry *
2  __d_lookup(struct dentry * parent,
3             struct qstr * name)
4  {
5    unsigned int len = name->len;
6    unsigned int hash = name->hash;
7    const unsigned char *str = name->name;
8    struct hlist_head *head =
     ➥d_hash(parent,hash);
9    struct dentry *found = NULL;
10   struct hlist_node *node;
11
12   rcu_read_lock();
13   hlist_for_each (node, head) {
14     struct dentry *dentry;
15     unsigned long move_count;
16     struct qstr * qstr;
17
18     smp_read_barrier_depends();
19     dentry = hlist_entry(node, struct dentry,
20                          d_hash);
21     if (unlikely(dentry->d_bucket != head))
22       break;
23     move_count = dentry->d_move_count;
24     smp_rmb();
25     if (dentry->d_name.hash != hash)
26       continue;
27     if (dentry->d_parent != parent)
28       continue;
29     qstr = dentry->d_qstr;
30     smp_read_barrier_depends();
31     if (parent->d_op &&
32         parent->d_op->d_compare) {
33       if (parent->d_op->d_compare(parent, qstr,
34                                   name))
35         continue;
36     } else {
37       if (qstr->len != len)
38         continue;
39       if (memcmp(qstr->name, str, len))
40         continue;
41     }
42     spin_lock(&dentry->d_lock);
43     /*
44      * Ha a dentryt áthelyezték,
        ➥a keresés sikertelen
45      */
46     if (likely(move_count ==
47                dentry->d_move_count)) {
48       if (!d_unhashed(dentry)) {
49         atomic_inc(&dentry->d_count);
50         found = dentry;
51       }
52     }
53     spin_unlock(&dentry->d_lock);
54     break;
55   }
56   rcu_read_unlock();
57   return found;
58 }
```



7. ábra VFS csatolási fa

vfsmount ilyen formán történő összekapcsolása lehetővé teszi, hogy egy kicsit elegánsabban kezeljük az azonos befűzési ponthoz rendelt befűzéseket.

A 2. ábrán bemutatott példafájlrendszer a *7. ábrán* bemutatott `struct vfsmount` szerkezetet eredményezné. A `vfs1` szerkezetben az `mnt_mountpoint` és az `mnt_root` helyen egyarán az `r1` gyökér `dentry`-t találjuk, mivel ez lesz egyben a fájlrendszer legvégső gyökere is. A `vfs2` szerkezetben a `b` `dentry` `mnt_mountpoint` hivatkozásként szerepel, az `r2` pedig mint a hozzá tartozó `mnt_root`. Így amikor a `mount_hashtable` keresés visszaadja a `vfs2`-höz tartozó mutatót, az `mnt_root` mező alapján gyorsan beazonosíthatjuk a befűzött fájlrendszer gyökerét. A befűzött fájlrendszerek teljes alakja az ***mnt_mount/mnt_child*** listákból látható. Ezeket a listákat a `copy_tree()` és néhány további függvény használja, miközben visszacsatolt befűzést készít (loopback mount), amelyhez a teljes útvonalnévtér adott alfájában befűzött, valamennyi fájlrendszer áttekintése szükséges.

RCU alkalmazása a dcache-en

A `dcache` teljes párhuzamosítása meglehetősen összetett feladat volna, és úgy gondolták, hogy túlságosan kockázatos lenne beletenni a 2.5 változatba. A 2.6-os `dcache` mindössze az első lépést tette meg az RCU útján; a 2.7-es feladata lesz bármiféle zár foglalása nélkül a teljes utat bejárni.

Az útvonalszakasz-kereséseket az *1. listában* bemutatott `__d_lookup()` függvény végzi. Az `__d_lookup()` függvényt a szülőkönyvtár `dentry`-bejegyzését jelölő mutatóval és a keresendő névvel kell meghívni. A nevet `qstr` szerkezetben adjuk át, amelyben a karaktersorozatot megadó mutatót, a szöveg hosszát, a `dcache` hasítótáblához használható előfeldolgozott hasítóértéket, valamint – amennyiben szükséges – magát a nevet találjuk.

Az 5–7. sor a `struct qstr` szerkezetet értelmezi. A 8. sor a név és a szülő `dentry` mutató kombinációját rendeli egy általános `dcache` hasítótáblához, kinyerve belőle a megfelelő hasítólánchoz tartozó mutatót.

A 12. és 56. sor jelöli a kód RCU-védett szakaszait, kikapcsolva az időosztást (preemptive) a `CONFIG_PREEMPT`

2. lista Útvonal–név–szakaszkeresés és -átnevezés versenyhelyzetének feloldása

```
 1 struct dentry *
 2 d_lookup(struct dentry * parent,
 3          struct qstr * name)
 4 {
 5   struct dentry * dentry = NULL;
 6   unsigned long seq;
 7
 8   do {
 9     seq = read_seqbegin(&rename_lock);
10     dentry = __d_lookup(parent, name);
11     if (dentry)
12       break;
13   } while (read_seqretry(&rename_lock, seq));
14   return dentry;
15 }
```

3. lista A dentry-szerkezetek késleltetett felszabadítása

```
1 static void d_free(struct dentry *dentry)
2 {
3   if (dentry->d_op && dentry->d_op->d_release)
4     dentry->d_op->d_release(dentry);
5   call_rcu(&dentry->d_rcu, d_callback,
    ➥dentry);
6 }
```

4. lista A dentry-bejegyzésekhez tartozó RCU callback függvény

```
1 static void d_callback(void *arg)
2 {
3   struct dentry * dentry = (struct dentry
    ➥*)arg;
4
5   if (dname_external(dentry)) {
6     kfree(dentry->d_qstr);
7   }
8   kmem_cache_free(dentry_cache, dentry);
9 }
```

rendszermagokban, csakúgy, mint a Linux Journal 2003. októberi számában olvasható „Using RCU in the Linux 2.5 Kernel” cikkben bemutatott Reader-Writer-Lock/RCU esetében. A 13–55. sor a kiválasztott hasítólánc elemein lépked végig megfelelő `dentry` után kutatva. A 18. sor egy, csak a DEC Alpha gépeken lényeges memóriakorlát kezelését végzi. Egyéb processzorokon a mutató visszakódolása által létrehozott adatfüggőség önmagában elegendő, így ezeken a processzorokon a 18. sor nem készít kódot. Minthogy ez a keresés nem kér zárat, versenyhelyzetbe kerülhet a `rename` (átevezés) rendszerhívással. Egy ilyen rendszerhívás a `dentry`-t egy másik hasítóláncba mozgathatja át, magával rántva az egész keresést. A 21. és 22. sor az ilyen versenyhelyzetek létét ellenőrzi, de önmagában még nem elegendő. Ezért a 23. sor megjegyzi azt a pillanatot (pillanatfelvételt készít), amikor a jelenlegi `dentry` a `dcache d_move()` függvény átnevezési folyamata alá került, így később könnyen meg lehet állapítani, hogy volt-e olyan átnevezés, amely az útvonalsétával versenyhelyzetbe került volna. A 24. sor memóriagátja arra szolgál, hogy a pillanatfelvételt se a fordító, se a CPU ne rendezhesse újra.

A 25–28. sor a szülő `dentry`-t és a név-hasht ellenőrzi. Amennyiben bármelyik hibásnak bizonyulna, ez a `dentry` nem lehet a keresésünk alapja. A 29–41. sor végzi a teljes névösszehasonlítást a DEC Alpha gépekhez szánt memóriagáttal (30. sor). A 33. sorban láthatjuk, hogy alkalmazhatunk-e fájlrendszerfüggő névösszehasonlító függvényeket is (például a kis- és nagybetűfüggetlen fájlrendszerekhez). Amint a végrehajtás eléri a 42. sort, megtaláltuk a megfelelő névhez tartozó gyermek `dentry` bejegyzést. Most már elkérhetjük a gyermek `dentry` zárját (42. sor). Minthogy valamennyi `dentry`-bejegyzésen külön zárat alkalmazunk, ezeknek az egyedi záraknak a versengési szintje lényegesen alacsonyabb, mint az eredeti `dcache_lock` rendszeré volt. Sajnálatos módon az élet nem fenékig tejfel, ugyanis például a gyökér `dentry` lezárása továbbra is versengéshez vezet – de ezzel majd később foglalkozunk.

Elképzelhető, hogy a gyermek `dentry`-t már átnevezték azóta, hogy a 23. sorban meghívtuk a `d_move_count` pillanatkép készítését. Ezért a 46–47. sor a `d_move_count` és a pillanatkép értékét hasonlítja össze. Ha az ellenőrzés egyezést mutat, a gyermek `dentry`-t nem nevezték át a keresés ideje alatt, és a 48–51. sor növeli a számlálót – de csak akkor, ha a bejegyzés még mindig hashelt.

Az 53. sor elengedi a gyermek `dentry` zárát, az 54. pedig kilép a hasítólánc keresési ciklusából. Az 57. sor sikeres keresés esetén a mutatót, sikertelen esetben pedig a `NULL` értéket adja vissza a gyermek `dentry`-nek.

Az `__d_lookup()` függvény sikertelensége nem feltétlenül jelenti azt, hogy a felhasználói folyamat is sikertelenséget jelző üzenetet kap. A fájl ettől még létezhet, csak éppen még nincsen betöltve a gyorstárba.

A függvény azonban nem oltalmaz meg bennünket valamennyi átnevezési verseny kockázatától. Versenyhelyzet jöhet létre ugyanis abban az esetben is, amikor a `dcache` listák (list) helyett `hlist` szerkezetet használ a `dcache` hasítóláncokhoz. A `hlist` listákat memóriamegtakarítás céljából alkalmazza, minthogy a `hlist` listák kettő helyett csak egyetlen mutatót tárolnak a lista fejlécében. Ez egyúttal azt is jelenti, hogy a listákkal ellentétben a `hlist` nem körkörös. Ezért előfordulhat, hogy egy adott `dentry`-t úgy nevezünk át, hogy az egy korábban üres `dcache` hasítóláncba kerül. Ha ez éppen az adott időben történik, az `__d_lookup()` függvény (helytelen módon) keresési hibát ad vissza.

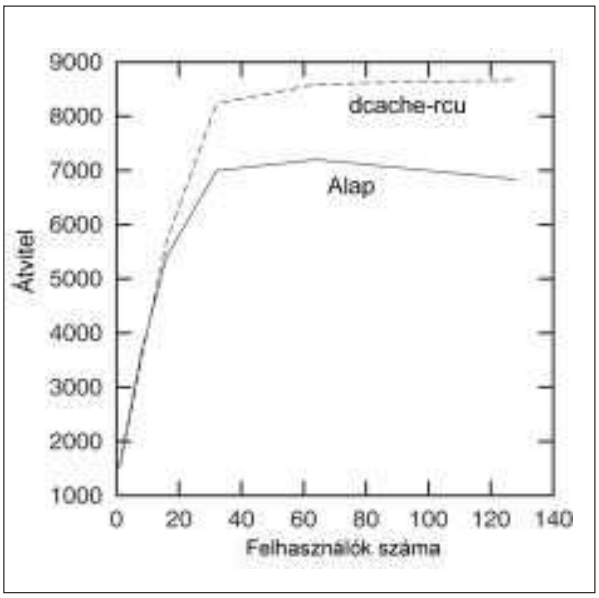
A hibásan visszaadott keresési eredménytelenséget a felsőbb szintű `d_lookup()` függvény kezeli, amit a *2. listán* mutatunk be. A versengő átnevezéseket a `read_seqretry()` függvény érzékeli a 13. sorban. Mivel a gondot okozó eset kizárólag hamis hibajelentést küldhet, az ellenőrzést elegendő csak az `__d_lookup()` `NULL` visszatérési értékeire elvégeznünk.

5. lista A dentry-bejegyzések átnevezése

```
 1 void
 2 d_move(struct dentry *dentry,
 3         struct dentry *target)
 4 {
 5   spin_lock(&dcache_lock);
 6   write_seqlock(&rename_lock);
 7   if (target < dentry) {
 8     spin_lock(&target->d_lock);
 9     spin_lock(&dentry->d_lock);
10   } else {
11     spin_lock(&dentry->d_lock);
12     spin_lock(&target->d_lock);
13   }
14   if (dentry->d_vfs_flags & DCACHE_UNHASHED)
15     goto already_unhashed;
16   if (dentry->d_bucket != target->d_bucket) {
17     hlist_del_rcu(&dentry->d_hash);
18 already_unhashed:
19     dentry->d_bucket = target->d_bucket;
20     hlist_add_head_rcu(&dentry->d_hash,
21                        target->d_bucket);
22     dentry->d_vfs_flags &= ~DCACHE_UNHASHED;
23   }
24   __d_drop(target);
25   list_del(&dentry->d_child);
26   list_del(&target->d_child);
27   switch_names(dentry, target);
28   smp_wmb();
29   do_switch(dentry->d_name.len,
30             target->d_name.len);
31   do_switch(dentry->d_name.hash,
32             target->d_name.hash);
33   if (IS_ROOT(dentry)) {
34     dentry->d_parent = target->d_parent;
35     target->d_parent = target;
36     INIT_LIST_HEAD(&target->d_child);
37   } else {
38     do_switch(dentry->d_parent,
39               target->d_parent);
40     list_add(&target->d_child,
41        &target->d_parent->d_subdirs);
42   }
43   list_add(&dentry->d_child,
44      &dentry->d_parent->d_subdirs);
45   dentry->d_move_count++;
46   spin_unlock(&target->d_lock);
47   spin_unlock(&dentry->d_lock);
48   write_sequnlock(&rename_lock);
49   spin_unlock(&dcache_lock);
50 }
```

Halasztott felszabadítás

A `d_free()` függvénynek egy megadott türelmi ideig el kell halasztania az adott `dentry`-hez tartozó területek felszabadítását, mivel azon a bejegyzésen éppen tetszőleges



8. ábra Többfelhasználós teljesítménymérés eredménye

számú folyamatban lévő útvonalséta tarthat mutatókat. A `d_free()` függvényben alkalmazott halasztást a *3. listában* tekinthetjük meg, ahol az 5. sor a `call_rcu()` primitív segítségével késlelteti a `d_callback()` pusztítóműveleteket, amíg egy adott türelmi idő le nem telik. A `d_callback()` függvény listáját a *4. lista* tartalmazza; ez – amennyiben szükséges – a külön tárolt fő részeket (5–7. sor) egyszerűen felszabadítja, majd a 8. sorban magát a `dentry`-t is felszabadítja.

Átnevezés

A `d_move()` függvény valósítja meg az (átnevezést végző) `rename` rendszerhívás `dentry`-re jellemző elemeit, ezt mutatja be *5. listánk*. Az 5. sor kizár minden, esetlegesen `dcache` frissítést végező egyéb folyamatot, majd a 6. sor lehetővé teszi a `d_lookup()` függvénynek, hogy megállapítsa, valóban versenyzett-e valamilyen átnevezéssel.
A 7–13. sor az átnevezés alatt lévő állomány és annak célja számára foglal le `dentry` elemenként egy-egy zárat, mégpedig a holtzárak (deadlock) elkerülése érdekében, cím szerint rendezve. A 14–17. sor eltávolítja a bejegyzést a `dcache` hasítótáblában elfoglalt régi helyéről, amennyiben az korábban nem törlődött.
A 19. sor frissíti a `dentry` bejegyzést, hogy az az új hasítógyűjteményre mutasson; a 20–21. sor felveszi a `dentry`-t a hozzá tartozó cél-hasítógyűjteménybe, végül a 22. sor frissíti a jelzőket (flag), mutatván, hogy a `dentry` a `dcache` hasítótáblában található meg. A 24. sor eltávolítja az (általunk átnevezett) cél-`dentry`-t a `dcache` hasítótáblából, a 25–26. sor pedig a mozgató és a cél-`dentry`-ket választják el a régi szülőjüktől.
A 27. sor megváltoztatja a `dentry` nevét, a 28. sor pedig kikényszeríti az írás újrarendezését. A névváltoztatás megoldása nem olyan egyszerű, mivel a rövid neveket magában a `dentry`-ben tároljuk, a hosszabb nevek viszont külön lefoglalt memóriahelyre kerülnek. A 29–32. sor frissíti a névhosszakat és hasítóértékeket. A 33–44. sor

a dentry-t csatlakoztatja annak új szülőjéhez. Végül a 45. sor frissíti a d_move_count értéket, így az __d_lookup() felfigyelhet a versenyhelyzetre; végül a 46–49. sor kioldja a zárat.
Elméletben tetszőlegesen szerencsétlen keresést készíthetnénk, a dentry értékeiket mindig azonos könyvtárban és azonos hasítóláncban hagyó, gondosan megtervezett átnevezési műveletek folyamatos sorozatával. Az egyik lehetőség, amikor ilyesmi bekövetkezhet, ha a hasítólánc utolsó elemét keressük, miközben az utolsó előtti elemet folyamatosan átnevezzük (és az ekként mindig a lista elejére kerül), épp mire a keresés odaérne. A gyakorlatban azonban a dcache hasítóláncok rövidek, az átnevezés pedig lassú. Amennyiben az ilyen elszállások gondot jelentenek, elképzelhető, hogy ki kell egészíteni a kódot, és útvonalbejárási sikertelenség esetén az átnevezést le kell állítani. Egy másik lehetséges megoldás szerint teljes egészében el kell távolítani globális hasítótáblát, és helyette a d_subdirs listákat szükséges olyan módon megváltoztatni, hogy a nagy könyvtárakat elegánsan kezeljék.

Teljesítmény és összetettség összehasonlítása

Bár a dcache-kódban végzett változtatás hozzávetőlegesen kicsi volt, igen távoli következményei is vannak a rendszermagban, ugyanis ez idáig nem létezett a dcache-sel együttműködő, jól körülhatárolt fájlrendszer-API. Ennek eredményeképpen rengeteg hiba bukkant fel a Linux 2.5-ös rendszermagban, mivel a fájlrendszerkódolók a dcache-t hagyományos stílusban, közvetlenül próbálták meg módosítani. Tekintve, hogy immár létezik egy formálisabb API, remélhetjük, hogy a következő változtatások már kevésbé lesznek sokkolóak.
A *8. ábrán* egy többfelhasználós tesztprogram teljesítményét láthatjuk az RCU-folttal ellátott Linux 2.5.59-es rendszermag és a folt nélküli mag esetében. A próbákat 16 processzoros NUMA-Q rendszeren futtattuk, 700 MHz-es Pentium III Intel Xeon processzorok felhasználásával, 1 MB L2 gyorstárral és 16 GB memóriával.
A dcache_rcu folt alkalmazása a Linux 2.4.17 rendszermagra 2,258-ról 2,530-ra növelte a SPECweb99 (SSL nélküli) átviteli teljesítményét egy nyolcprocesszoros Pentium III Xeon kiszolgálón, ami 12 százalékos gyorsulást jelent. Ugyanezt a foltot alkalmazva a Linux 2.5.40-mm2 rendszermagra a Linux-rendszermag fordítási ideje 47,548 CPU-másodpercről 42,498 CPU-másodpercre esett vissza, ami több mint tízszázalékos gyorsulást jelent. Hasonló próbát futtatva a Linux 2.5.42 rendszermagot használó egyprocesszoros 700 MHz-es Pentium III Xeon gépen nem tapasztaltunk változást. Összefoglalva tehát: a dcache RCU nemcsak, hogy növeli a felső kategóriás gépek méretezhetőségét, de gyakran az alacsonyabb kategóriás gépek teljesítményét is fokozza.

A jövő irányvonalai

Bár a 2.6 dcache rendszer sokkal méretezhetőbb, mint a 2.4-es változat volt, számos dolog még vizsgálatra szorul:

1. A frissítések felett továbbra is a dcache_lock őrködik, így a mélyreható frissítést használó folyamatok nem jól méreteződnek.
2. A globális hasítótábla miatt a gyorstár nem lehet helyi és a frissítési kód a szükségesnél összetettebb lesz. Természetesen az alternatíváknak meg kell tartaniuk az előd vívmányait, többek közt a nagy könyvtárak nagyteljesítményű kezelését.
3. A 2.6-ban lévő dcache-kód minden egyes dentry d_lock spinlock zárját lekérdezi, ami gyorstárvonalon pattogáshoz (bouncing) és atomi műveletvégzéshez (atomic operations) vezethet, különös tekintettel a gyökérkönyvtárra és a munkakönyvtárakra. Még sokat kell gondolkodni egy igazán egyszerű megoldáson, ugyanis az engedélyeknek a dentry-bejegyzésekbe történő átmozgatása túlságosan bonyolultnak bizonyult.
4. Az __d_lookup() és a d_move() közötti versenyhelyzetek feloldására használt kód túlzottan bonyolult.

Izgatottan várjuk, hogy a 2.7 fejlesztései megoldják ezeket a kérdéseket.

Jogi megállapítások

Az itt leírtak a szerző álláspontját tükrözik, és nem feltétlenül egyeznek meg az IBM véleményével. A SPEC és a SPECweb név a Standard Performance Evaluation Corporation bejegyzett védjegye. A teljesítménymérés kizárólag kutatási célokat szolgált és a szabályoktól való következő eltérések miatt nem hasonlítható össze a SPECWeb lapján található eredményekkel:

1. Olyan alkatrészeken futott, amik nem felelnek meg a SPEC-nagyközönség számára elérhetőnek, a gép ugyanis mérnöki mintadarab volt.
2. Az access_log naplót nem tartottuk meg bizonyítékként. Kiírattuk ugyan, de 200 másodpercenként le is töröltük.

A legfrissebb SPECweb99-teljesítménypróbák a
➲ http://www.spec.org lapon érhetők el.

Linux Journal 2003. február, 117. szám

Paul E. McKenney
Kiváló mérnök az IBM-nél. Régebb óta dolgozik az SMP és NUMA algoritmusokon, mint hogy érdemesnek tartaná megemlíteni. Ezt megelőzően a packet-radio és internetes protokollok témájával foglalkozott (jóval azelőtt, hogy az internet népszerű lett). Hobbijai közé tartozik a futás és a szokásos „ház–asszony–gyerekek" álom.

Dipankar Sarma
Jelenleg több Linux-rendszermagprojekten is dolgozik, többek közt a RCU és VFS fejlesztéseken. Linuxos kora előtt számos egyéb területen dolgozott, többek közt az ABI, OS felhozatal, B-K-meghajtók és többösvényű B-K témájában.

Maneesh Soni
Az IBM Linux Technology Központban dolgozik a Linux méretezhetőségének fejlesztési projektjén. Nagy tapasztalatokkal rendelkezik a rendszerprogramok, különösen az operációs rendszermagok és a fájlrendszerek területén.

B-K-ütemezők

Nézzük meg, milyen hatással lesz az új B-K-ütemező megjelenése a Linux lemezteljesítményére, illetve milyen előnyöket várhatunk a 2.6-os rendszermagban megjelent B-K-ütemezőtől.

A legtöbb Linux-felhasználó tisztában van a folyamatütemező szerepével (jó példa erre az új O(1) ütemező), azonban viszonylag kevesen ismerik az B-K (bevitel/kivitel) ütemezők működését. A B-K-ütemezők bizonyos szempontból hasonlítanak a folyamatütemezőkre: mind a kettő erőforrásokat ütemez több felhasználó között. A folyamatütemező a rendszeren futó folyamatoknak kiszabott processzoridőért felelős. Mit ütemez akkor a B-K-ütemező?

Egy rendszerben nem is feltétlenül szükséges a jelenléte, vagyis a folyamatütemezővel ellentétben a B-K-ütemező nem elengedhetetlen része az operációs rendszernek.

Az B-K-ütemező egyedüli „raison d'□tre"-je a teljesítmény. Mivel meg szeretnénk érteni a B-K-ütemezők szerepét, át kell rágnunk magunkat néhány háttéradaton, ezt követően megnézzük, miként viselkedne egy B-K-ütemező nélküli rendszer. A merevlemezek az ismerős sáv, fej, szektor felosztású, geometriai alapú címzéssel kezelik az adatokat. A merevlemez több tálcából áll, ezek mindegyike tartalmaz egy-egy lemezt, tengelyt és író–olvasófejet. Minden tálca – a CD-khez hasonlóan – további körkörös, gyűrűszerű pályákra bontható; végezetül minden ilyen sáv egész számú szektorból áll.

Ha a merevlemezen egy bizonyos adategységet keresünk, a meghajtó logikája három adatot vár tőlünk: a sáv, a fej és a szektor számát. A sáv mondja meg, hogy melyik körkörös pályán találjuk az adatot. Ha a tálcákat egymás fölé helyezzük (miként a merevlemezben is elhelyezkednek), a megadott sáv egy hengert alkot a tálcákon keresztül (innen származik az angol elnevezése: cylinder, azaz henger). A kérdéses író–olvasófejet (és ezzel egyúttal a tálcát) a fejadat adja meg. A keresést ezáltal egyetlen tálca egyetlen sávjára szűkítettük le. Végül a szektorérték azonosítja a sávunk adott szektorát. A keresés ezzel befejeződött: a merevlemez már tudja, hogy az adat melyik szektorban, melyik sávon és melyik tálcán helyezkedik el. Beállíthatja az író–olvasófejet a megfelelő tálca megfelelő sávja fölé és beolvashatja a megfelelő szektort.

Szerencsére a korszerű merevlemezek már nem kényszerítik a gépeket arra, hogy sáv-, fej- és szektoralapon tartsák velük a kapcsolatot. A jelenlegi merevlemezek minden egyes sáv–fej–szektor hármashoz egyedi blokkszámot rendelnek. A sáv–fej–szektorértéket ez az egyedi azonosító adja meg. A mostani operációs rendszerek e blokkszám használatával tarthatják a kapcsolatot a merevlemezzel (ezt nevezik logikai blokkcímzésnek), amit aztán a merevlemez a helyes sáv–fej–szektorértékre fordít le magának. Ezzel a blokkos címzéssel kapcsolatban egy dolgot érdemes megemlíteni: bár garancia nincs rá, a fizikai hozzárendelés általában sorban történik, vagyis az n-edik logikai blokk fizikailag többnyire az n+1-dik logikai blokk után következik. Hogy ez miért fontos, arról később még szó lesz.

Most nézzünk meg egy átlagos Unix-rendszert: blokkokat olvasó vagy kiíró B-K-kérelmeket a lemezen alkalmazások széles kínálata, adatbázisok, levelezőügyfelek, webkiszolgálók és szövegszerkesztők használnak. A blokkok fizikailag általában összevissza helyezkednek el a lemezen. A postaláda adatai teljesen más helyen lehetnek, mint a webkiszolgáló HTML-adatai vagy a szövegszerkesztő beállításfájljai. Igazság szerint létezhet olyan állomány is, amely fizikailag az egész lemezen található, amennyiben szét van darabolva, azaz nem egymást követő blokkokban helyezkedik el. Mivel a fájlokat egyedi blokkokba tördeltük, a merevlemez pedig blokkokat kezel és nem a jóval elvontabb fájlokat, az állomány adatainak olvasását több, különféle blokkokra hivatkozó egyedi B-K-kérelem sorozatára kell

1. próba: Írás miatt koplal az olvasás

Egy folyamatos írást futtassunk a háttérben, például:

```
while true
do
        dd if=/dev/zero of=file bs=1M
done
```

Mindeközben mérjük meg, mennyi időbe telik egy 200 MB méretű fájl beolvasása:

```
time cat 200mb-file > /dev/null
```

Ez jól mutatja „az írás miatt koplal az olvasás" feladat nehézségét.

lebontani. Szerencsés esetben a blokkok sorban helyezkednek el vagy legalább fizikailag közel találhatóak egymáshoz. Amennyiben a blokkok nincsenek közel egymáshoz, a lemez fejének egy másik helyre kell mozdulnia a lemezen. A lemezfej mozgatását keresésnek (seeking) nevezik, és a számítógépen ez az egyik legidőigényesebb művelet. A mai merevlemezek keresési ideje tíz milliszekundumokban mérhető. Ez az egyik oka annak, amiért jó dolog, ha az állományaink töredezettségmentesek (defragmented). Sajnos azonban nem igazán számít, hogy fájljaink töredezetlenek-e vagy sem, mivel a rendszer az egész lemezen egyszerre több állományhoz hoz létre B-K-kérelmeket.
A levelezőügyfél egy kicsit innen szeretne, a webkiszolgáló egy kicsit onnan… De várjunk csak, a szövegszerkesztő épp most szeretne beolvasni egy fájlt. Ez együttesen annyit jelent, hogy a meghajtófej összevissza ugrál a lemezen.
A legrosszabb esetben több állomány együttes B-K-kérelmeinél a fej egyfolytában csak ide-oda ugrál, ez pedig nincs valami jó hatással a rendszerteljesítményre.
Ezen a ponton lép be a képbe a B-K-ütemező. Az B-K-ütemező ütemezi a függőben lévő B-K-kérelmeket, a legkisebbre véve a lemezfej mozgatására fordított időt. Ennek megfelelően csökken a keresési idő és növekszik a merevlemez teljesítménye.
A varázslat két fő feladat: a rendezés és egyesítés segítségével hajtható végre. Először is a B-K-ütemező egy blokkszám szerint rendezett listát tart fenn a függőben lévő B-K-kérelmekről. Ha új B-K-kérelem érkezik, blokkszám szerint be kell szúrni a függőben lévő kérelmek listájára, így a meghajtófejnek nem kell összevissza szaladgálnia a lemezen, hogy a B-K kérelmeket kiszolgálhassa. Amíg a listát rendezetten tartjuk, a lemezfej szépen sorban halad a lemezen. Ha a meghajtó éppen egy kérelmet szolgál ki a lemez valamelyik részén és pontosan erre a területre vonatkozó új kérelem érkezik, akkor ezt a kérelmet ki lehet szolgálni, még mielőtt egy másik helyre mozgatnánk a fejet.
Az egyesítést olyankor használjuk, amikor az azonos vagy sorrendben következő lemezrészre hivatkozó B-K-kérelem érkezik. Új, önálló kérelem kiadása helyett a kérelmet egyszerűen egyesítjük az azonos vagy következő kérelemmel – ezáltal a legkisebbre csökkenthetjük a várakozó kérelmek számát.
Lássunk egy példát! Képzeljük el azt az esetet, amikor két alkalmazás a felsorolt blokkszámokra ad ki kérelmeket, és ezek a következő sorrendben érkeznek a rendszermaghoz: 10, 500, 12, 502, 14, 504 és 12. A B-K-ütemező nélküli megközelítés a blokkokat a megadott sorrendben szolgálná ki, ami hét hosszú keresést jelentene, oda-vissza a lemez két különböző területe közt. Micsoda pazarlás! Ha a rendszermag a kérelmeket sorba rendezte és egyesítette volna, és ilyen módon szolgálta volna ki őket, ez az eredmény egészen más lenne: 10, 12, 14, 500, 502 és 504. Mindössze egyetlen távoli keresés és összességében is eggyel kevesebb kérelem. Ezzel a módszerrel a B-K-ütemező több B-K-kérelem számára teszi elérhetővé a lemezforgalom erőforrását, maximalizálva a teljes átvitelt. Minthogy a B-K-átvitel meglehetősen fontos a rendszer teljesítménye szempontjából és mivel a keresések ilyen iszonyatosan lassúak, a B-K-ütemező munkája nagyon fontos.

Az eredmények

B-K-ütemező és rendszermag	1. teszt	2. teszt
Linus lift 2.4 alatt	45 s	30 perc, 28 s
Határidős B-K-ütemező 2.6 alatt	40 s	3 perc, 30 s
Jósló B-K-ütemező 2.6 alatt	4,6 s	15 s

2. próba: A nagy olvasási lappangás hatásai

Indítsunk folyamatos olvasást a háttérben:

```
while true
do
        cat big-file > /dev/null
done
```

Mindeközben mérjük meg, hogy mennyi ideig tart beolvasni a rendszermagfa valamennyi állományát:
`time find . -type f -exec cat ´{}´ ´;´ > /dev/null`
Így kipróbálhatjuk, hogyan viselkedik egyetlen nagy folyamatos olvasás alatt végrehajtott sok apró függő olvasás.

A Linus-lift

A 2.4-es Linux-rendszermagban található B-K-ütemezőt Linus liftnek (Linus Elevator) nevezik. A B-K-ütemezőket gyakran liftalgoritmusoknak nevezik, mivel hasonló feladatokkal birkóznak meg, mintha egy nagy épületben folyamatosan működtetni szeretnénk a liftet. A Linus-lift megoldás csaknem pontosan a fent leírt klasszikus B-K-ütemezőmodellt követi. A legtöbb szempontból ez kiválóan megfelelt, hiszen az egyszerűség jó dolog és a 2.4-es rendszermag B-K-ütemezője valóban jól működött. Sajnos a B-K-ütemező teljes B-K-átvitel maximalizálását célzó küldetése során engedményt kellett tenni, az ár pedig a méltányosság sérülése lett (különösen kérelemlappangás – latency – esetén). Lássunk egy példát!
Vegyük a logikai lemez blokk-kérelmeinek a következő sorozatát: 20, 30, 700 és 25. A B-K-ütemező rendező algoritmusa a következő sorrendbe rendezné és adná ki a kérelmeket (feltételezve, hogy a fej jelenleg a lemez logikai kezdőpontjában áll): 20, 25, 30 és 700. Ez az, amire számítottunk, eddig nincs is gond. Tegyük fel azonban, hogy a 25-ös blokk kiszolgálása közben egy újabb kérés érkezik a lemez azonos részére. Azután egy újabb. Majd egy még újabb. Igencsak valószínű, hogy a várólista 700-as blokkra hivatkozó kérelme egy jó darabig nem lesz kiszolgálva.
De ez még nem minden. Mi történik, ha a kérelem egy lemezblokk beolvasása volt? Az olvasási kérelmek általában összehangoltak. Amikor az alkalmazás lemezírási utasítást ad ki, általában megáll és várakozik, amíg a rendszermag a kért adatot vissza nem adja. Az alkalmazásnak üldögélnie kell és várakozás közben malmozhat, míg a 700-as blokk kérelmére végre válasz érkezik. Az írás ezzel szemben általában nem összehangolt, hanem aszinkron módon működ-

dik. Amikor az alkalmazás kiad egy írási műveletet, a rendszermag az írandó adatot és a metaadatot bemásolja a rendszermagba, felkészít egy vermet az adat számára és visszalép az alkalmazáshoz. Az alkalmazást nem érdekli és igazából nem is kell tudnia, hogy az adata ténylegesen mikor kerül a lemezre.

Ettől viszont olvasási kérelem barátunk helyzete csak tovább romlik. Mivel az írás aszinkron, általában folyamként viselkedik, azaz gyakori a nagy mennyiségű adat egy idejű visszaírása. Ennek következtében rengeteg egyedi írási kérelem kerül a merevlemez közeli területeire. Példaképpen képzeljük el egy nagy állomány mentését: az alkalmazás írási műveleteket küldözget a rendszernek, a merevlemez pedig olyan gyors, amilyen gyorsra ütemezzük.

Az olvasási kérelmek ezzel szemben általában nem folyamként viselkednek. Az alkalmazások az olvasási kérelmüket apró egyenkénti darabokban adják ki, ahol is minden darabka az utolsótól függ. Gondoljunk csak egy könyvtár bejegyzéseire: az alkalmazás megnyitja az első állományt, kiadja az olvasási kérelmet az állomány megfelelő szakaszára, megvárja a visszaadott adatot, kiadja a következő darab olvasási igényét, vár, majd így folytatja mindaddig, amíg az egész állományt be nem olvassa. Ezután az állományt bezárja, megnyitja a következőt – és a folyamat kezdődik elölről. Minden soron következő kérelemnek meg kell várnia az előzőt, ami igen komoly késedelmet okozhat alkalmazásunk számára, ha a kérelmek túlságosan messzi lemezblokkokban találhatók.

Az a jelenség, amikor a folyamot alkotó írási kérelmek koplalásra ítélik a függő olvasási kérelmeket, „írás miatt koplal az olvasás” (writes-starving-reads) néven vált ismertté (lásd a széljegyzetet: „1. próba: Írás miatt koplal az olvasás”). Annak a lehetőségét, ha egy kérelmet elfogadható időn belül nem szolgálnak ki, éhezésnek nevezzük. A kérelmek éheztetése igazságtalansághoz vezet. A B-K-ütemezők esetében a rendszer a nagyobb átlagos átviteli teljesítmény érdekében egyértelműen feladta az igazságosságot. Azaz a rendszer megpróbálja növelni a rendszer összesített teljesítményét, bármely egyedi B-K-kérelem lehetséges feláldozása árán is. Ez elfogadott és tulajdonképpen kívánatos is, eltekintve attól, hogy a hosszúra nyúló éhezés nem valami jó dolog. Az olvasási kérelmek közepes ideig tartó éheztetése is nagy lappangási időhöz vezet, ha az alkalmazások más lemezműveletek közben próbálnak olvasási kérelmeket kiadni. Ez a nagy lappangás hátrányosan érinti a rendszer teljesítményét és kényelemérzetét (lásd a széljegyzetet: „2. próba: a magas olvasási lappangás hatásai”).

A határidős B-K-ütemező

Az új, 2.6-os B-K-ütemezők fő célja pontosan az ilyen éhezés elkerülése volt általában a kérelmek, különös tekintettel az olvasási kérelmek esetében.

A 2.4-es B-K-ütemező és általában véve a hagyományos liftalgoritmusok körüli nehézségek kezelésére jelent meg a határidős B-K-ütemező. Mint azt bemutattuk, a Linus-lift a függőben lévő B-K-kérelmek listáját egyetlen sorban tárolja. A soron következő kiszolgált kérelem a sor élén álló B-K-kérelem lesz. A határidős B-K-ütemező megtartja ezt a sort, de egy kicsit megfűszerezi a dolgokat két további sor – az olvasási FIFO és az írási FIFO sor – bevezetésével. A határidős B-K-ütemező mindkét sorban küldési idő szerint rendezve tárolja az elemeket (azaz lényegében az elsőként belépő távozik elsőként, ezt jelenti egyébként a FIFO rövidítés: first in, first out – a ford.). Az olvasási FIFO sor, mint a neve is sugallja, kizárólag olvasási kérelmeket tárol. Az írási FIFO sor hasonlóképpen kizárólag az írási műveletekkel foglalkozik. Minden FIFO sorhoz tartozik egy lejárati érték: az olvasási FIFO sor lejárati ideje 500 milliszekundum, az írási FIFO soré pedig öt másodperc.

Amikor új B-K-kérelmet kapunk, beszúrva a sorba rendezzük, valamint a megfelelő (olvasási vagy írási) FIFO sor végére helyezzük. Normál esetben a merevlemeznek a rendezett, hagyományos sor tetejéről küldjük a kérelmeket, ezáltal maximalizáljuk az átvitelt és minimalizáljuk a kereséseket, hiszen akárcsak a Linus-lift esetében, a normál lista blokkszám szerint rendezett.

Amikor az egyik FIFO sor elején álló elem idősebb lesz, mint a sorhoz rendelt lejárati idő, a B-K-ütemező felfüggeszti a normál sor B-K-kérelmeinek a továbbítását. Ehelyett a FIFO sor elején álló kérelmeket fogja elküldeni, az egyenletesség kedvéért még néhány további elemmel is megtoldva. A B-K-ütemezőnek csak a FIFO sorok elején álló bejegyzéseket kell ellenőriznie, hiszen mindig ezek lesznek a legöregebb elemek a sorban.

Emlékszünk még öreg barátunkra, a 700-as kért blokkra? A távoli blokkokat célzó írási kérelmek áradata ellenére 500 milliszekundum elteltével a határidős B-K-ütemező felfüggesztené az ottani kérelmek kiszolgálását és elküldené a 700-as blokk olvasási kérelmét. A lemez megkeresné a 700-as blokkot, kiszolgálná az olvasási kérelmet, majd folytatná a további várakozó kérelmek feldolgozását.

Ilyen módon a határidős B-K-ütemező lágy határidőt vezetett be a B-K-kérelmekre. Igaz, semmi sem biztosítja, hogy a B-K-kérelmek a lejárati idő előtt ki lesznek szolgálva, de a B-K-ütemező általában a lejárati időhöz közeli tartományon belül szolgálja ki a kérelmeket. Ennek megfelelően a B-K-ütemező továbbra is jó, teljes körű átvitelt szolgáltat, de mindeközben egyetlen kérelmet sem éheztet elfogadhatatlanul hosszú ideig. Mivel az olvasási kérelmekhez gyors lejárati időt adtunk meg, az „írás miatt koplal az olvasás” nehézségeit a lehető legkisebbre csökkentettük.

Jósló B-K-ütemező

Ez mind szép és jó, de még mindig nem a tökéletes megoldás. Gondoljuk végig, mi történik a képzeletbeli 700-as blokkunkkal, amely – tegyük fel – a lemezterületen számos egymástól függő olvasásnak az első tagja. Az olvasási kérelem kiszolgálása után a határidős B-K-ütemező a korábbi blokkokban folytatja az előzőleg elkezdett írási kérelmek kiszolgálását. Ez rendben is van egészen addig, amíg az olvasást végző alkalmazás a következő olvasási kérelmét el nem küldi (mondjuk a 710-es blokkra). 500 milliszekundumon belül ez a kérelem is lejár és a lemez a 710-es blokkhoz szalad, kiszolgálja a kérelmet, majd visszamegy oda, ahol az előbb állt és folytatja az írási kérelmek folyamának a kiszolgálását. Azután érkezik egy újabb olvasási kérés.

A feladat ismét azok miatt az átkozott függő írások miatt keletkezik! Mivel az olvasást függőben lévő darabok alapján végezzük, az alkalmazás csak akkor adja ki a következő olvasást, miután a korábbit már visszakapta. Csakhogy miközben az alkalmazás megkapja az adatot, a saját futási

üteméhez ér és elküldi a következő olvasást, a B-K-ütemező viszont már továbblépett és egy másik kérelmet szolgál ki. Az eredmény minden olvasáskor néhány felesleges keresés: kikeressük az olvasás helyét, kiszolgáljuk és visszamegyünk. Mennyivel jobb lenne, ha valami módja akadnak annak, hogy a B-K-ütemező megtudja – azaz inkább megjósolja –, hogy érkezik-e újabb olvasás a lemez azonos szakaszára. Előre-hátrakeresgélés helyett egyszerűen csak előrelátóan várna a következő olvasásra. A rút keresések kihagyása biztosan megér néhány milliszekundum várakozást és két keresést is kihagyhatunk.

Természetesen pontosan ezt teszi a jósló B-K-ütemező. Éppen úgy indul, mint a határidős B-K-ütemező: ugyanazt a határidő alapú rendszert használja, de ráadásképpen jósképességekkel is rendelkezik. Amikor az olvasási kérelem megérkezett, a jósló B-K-ütemező, akárcsak a korábbi, határidőn belül kiszolgálja. A határidős B-K-ütemezővel ellentétben azonban a jósló B-K-ütemező ezután csak üldögél és várakozik, semmit sem csinál egészen hat milliszekundumig. Igen jó esély van rá, hogy ez alatt a hat milliszekundum alatt az alkalmazás a fájlrendszer ugyanezen részére ad ki újabb olvasási kérelmet. Ha ez bekövetkezik, a kérelmet azonnal kiszolgáljuk, és a jósló B-K-ütemező még vár egy kicsit. Amennyiben a hat milliszekundum olvasási kérelem nélkül telik el, a jósló B-K-ütemező nem jól választott, és visszatér ahhoz, amit eddig csinált.

Mégha csak mérsékelt számú kérelmet sikerül is helyesen megjósolni, jelentős időt – darabonként két drága keresést – tudunk megtakarítani (lásd *táblázatunkat*). Minthogy a legtöbb olvasás függő, a jóslás a legtöbb esetben kifizetődő. Hogy a helyes jóslás esélyeit tovább növelje, a jósló B-K-ütemező heurisztikus módszereket alkalmaz, annak megállapítására melyik folyamatnak kell várnia. Ennek kivitelezéséhez az ütemező minden folyamathoz B-K-statisztikát vezet, nyomon követve annak viselkedését.

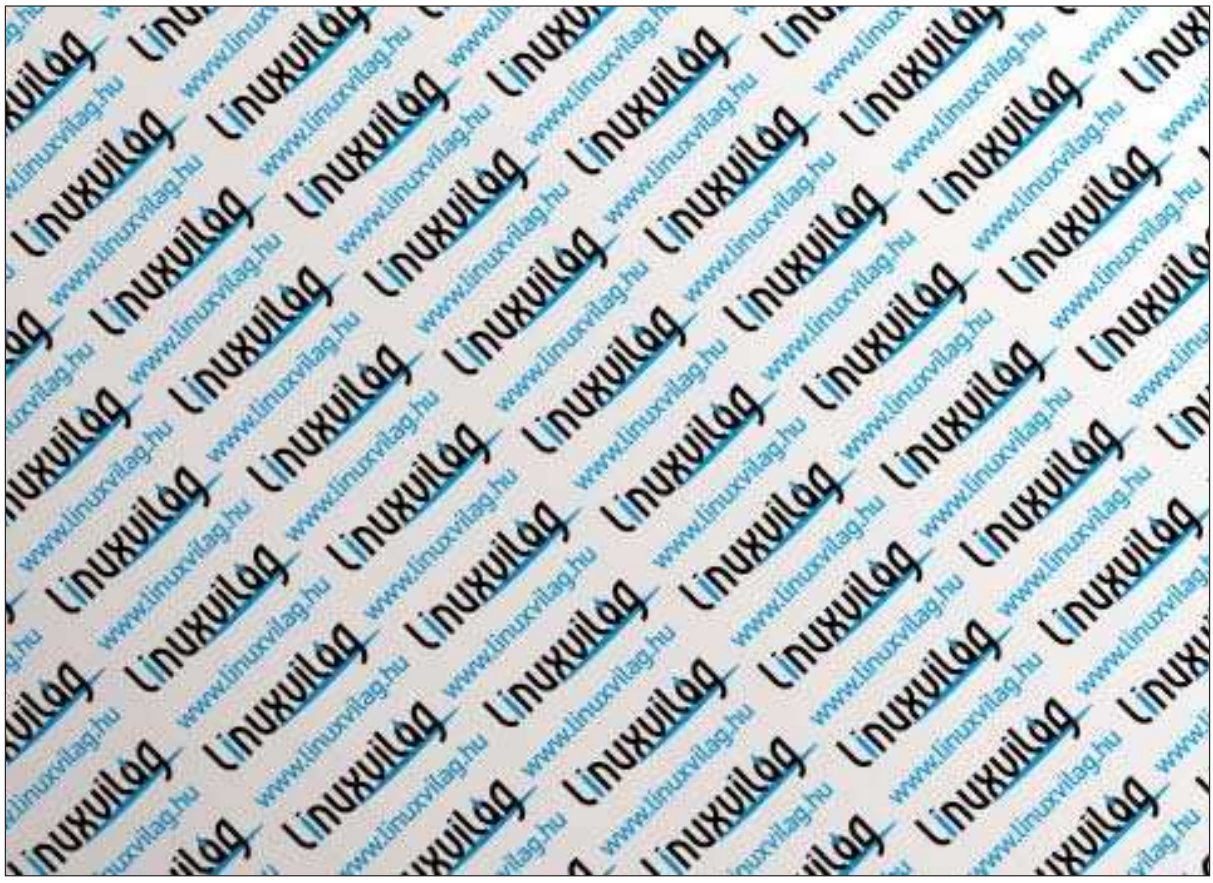
A szükségtelen keresések kiiktatásával és az olvasások gyorsabb kiszolgálásával a jósló B-K-ütemező egyszerre ad kisebb kérelemlappangási időt és nagyobb globális átvitelt a határidős B-K-ütemezőhöz vagy a Linus-lifthez képest. Nem meglepő, hogy a 2.6-os rendszermagban a jósló B-K-ütemező lett az alapértelmezett. Nagyon helyesen.

Köszönetnyilvánítás

A Linus-lift fő szerzője ***Andrea Arcangeli*** és ***Jens Axboe***. A határidős B-K-ütemező elsősorban Jens Axboe munkája, a jósló B-K-ütemező megszületése ***Nick Piggin*** érdeme.

Linux Journal 2004. február, 118. 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 punkzenét hallgat.



Aláírt magmodulok

A rendszermag ma már a beillesztésük előtt képes ellenőrizni a modulok titkosított aláírásait. Ennek részleteibe avatjuk be olvasóinkat.

Másik operációs rendszerekben már évek óta ismert az aláírással ellátott rendszermagmodulok módszere. Bizonyos emberek és cégek hasznosnak tartják, ha kizárólag olyan modulokat (avagy meghajtókat, ahogy néha nevezik) telepítenek a rendszerükre, amelyre operációs rendszerük valamelyik hitelesítője (authority) az áldását adta. A Linux-rendszermag modulbetöltési módszerének változását figyelembe véve az aláírt rendszermagmodulokat ma már könnyedén a rendszermagba illeszthetjük. Ez az írás arról szól, hogy miként valósítottam meg ezt az elképzelést, valamint azt is bemutatja, hogy milyen módon kell a gyakorlatban használni.

Az aláírt rendszermagmodulok lényege, hogy valaki digitális aláírást szúr be a modulba, jelezvén, hogy az adott modulban megbízik. Senkit sem akarok meggyőzni arról, hogy ennek benne kellene lennie a Linuxban vagy feltétlenül szükség van rá, sem arról, hogy növeli a biztonságot. Egyszerűen csak azt mesélem el, hogy miképpen lehet megvalósítani, és bemutatok egy megoldást, ha esetleg valaki használni szeretné.

Az aláírt rendszermagmodulok működéséhez nyílt kulcsú titkosítást használunk. Az RSA nyílt kulcsú titkosítási

```
$ readelf -S visor.ko
There are 23 section headers, starting at offset 0x3954:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        00000000 000040 0017e0 00  AX  0   0 16
  [ 2] .rel.text         REL             00000000 003cec 000cd0 08      21  1  4
  [ 3] .init.text        PROGBITS        00000000 001820 000210 00  AX  0   0 16
  [ 4] .rel.init.text    REL             00000000 0049bc 0001c8 08      21  3  4
  [ 5] .exit.text        PROGBITS        00000000 001a30 000030 00  AX  0   0 16
  [ 6] .rel.exit.text    REL             00000000 004b84 000030 08      21  5  4
  [ 7] .rodata           PROGBITS        00000000 001a60 000020 00   A  0   0 16
  [ 8] .rel.rodata       REL             00000000 004bb4 000028 08      21  7  4
  [ 9] .rodata.str1.1    PROGBITS        00000000 001a80 000449 01 AMS  0   0  1
  [10] .rodata.str1.32   PROGBITS        00000000 001ee0 0009c0 01 AMS  0   0 32
  [11] .modinfo          PROGBITS        00000000 0028a0 0006c0 00   A  0   0 32
  [12] .data             PROGBITS        00000000 002f60 000600 00  WA  0   0 32
  [13] .rel.data         REL             00000000 004bdc 0001e0 08      21  c  4
  [14] .gnu.linkonce.thi PROGBITS        00000000 003560 000120 00  WA  0   0 32
  [15] .rel.gnu.linkonce REL             00000000 004dbc 000010 08      21  e  4
  [16] __obsparm         PROGBITS        00000000 003680 000180 00  WA  0   0 32
  [17] .bss              NOBITS          00000000 003800 00000c 00  WA  0   0  4
  [18] .comment          PROGBITS        00000000 003800 00006e 00      0   0  1
  [19] .note             NOTE            00000000 00386e 000028 00      0   0  1
  [20] .shstrtab         STRTAB          00000000 003896 0000bd 00      0   0  1
  [21] .symtab           SYMTAB          00000000 004dcc 000760 10     22  58  4
  [22] .strtab           STRTAB          00000000 00552c 000580 00      0   0  1
```

algoritmus alapelveiről és működéséről a Linux Journal ➲ http://www.linuxjournal.com/article/6826 hálózati cikkében olvashatunk. Ez a cikk feltételezi, hogy az olvasó jártas a nyílt kulcsú titkosítás alapjaiban és képes egy új Linux-rendszermagot elkészíteni és betölteni a gépébe. Az új rendszermag elkészítéséről és betöltéséről bővebben a Linux Kernel HOWTO-ban (➲ http://www.tldp.org) olvashatunk.
A 2.5-ös fejlesztői rendszermagsorozatban ***Rusty Russell*** újraírta a Linux-magmodulok működését. A korábbi rendszermagokban a modulbetöltési logika nagyobbik része a felhasználói térben zajlott – Rusty változtatásai révén a teljes logika a rendszermagtérbe került; ilyen módon csökkent a kiépítésfüggetlen logika, valamint a felhasználói felület is lényegesen egyszerűsödött. Ennek egyik kellemes mellékhatásaként mostantól a rendszermag a teljes modulhoz hozzáférhet nyers formában. A magmodul egy egyszerű ELF formátumú program. Az ELF az executable and linking format (végrehajtható és csatoló formátum) szavak rövidítése és a végrehajtható programokat jelöljük vele. Az ELF szabványt a ➲ http://www.muppetlabs.com/~breadbox/software/ELF.txt címen találjuk meg szöveges formátumban.
Az ELF-állományok különböző szakaszokra bonthatók. Ezeket a szakaszokat tekinthetjük meg a `readelf` program futtatásakor, lásd az *1. listát*.
Minthogy az ELF-fájlok szakaszokból épülnek fel, nem túl nehéz egy újabb szakaszt felvenni a modul állományában, amit így modultöltés közben a rendszermag a memóriába olvas. Ha a modulba egy RSA-aláírásos szakaszt helyezünk,

```
for (i = 1; i < hdr->e_shnum; i++) {
    name = secstrings+sechdrs[i].sh_name;

    /* Csak azokkal a szakaszokkal foglalkozunk,
       ➥amelyeknek "text" vagy "data"
       olvasható a nevükben*/
    if ((strstr(name, "text") == NULL) &&
        (strstr(name, "data") == NULL))
        continue;
    /* a ".rel.*" szakaszok nem érdekelnek
       ➥bennünket. */
    if (strstr(name, ".rel.") != NULL)
        continue;

    temp = (void *)sechdrs[i].sh_addr;
    size = sechdrs[i].sh_size;
    do {
        memset(&sg, 0x00, sizeof(*sg));
        sg.page = virt_to_page(temp);
        sg.offset = offset_in_page(temp);
        sg.length = min(size,
                        (PAGE_SIZE - sg.offset));
        size -= sg.length;
        temp += sg.length;
        crypto_digest_update(sha1_tfm, &sg, 1);
    } while (size > 0);
}
```

a rendszermag visszakódolhatja az aláírást és összehasonlíthatja az éppen betöltött fájl aláírásával. Ha a kettő egyezik, az aláírás érvényes és a modul sikeresen beilleszthető a rendszermag memóriájába. Ha az aláírások nem egyeznek, akkor valaki vagy változtatott a modulon, vagy nem a megfelelő kulccsal írták azt alá. Ezután a modult elutasíthatjuk – erről szól ez a folt.

Hogyan működik a rendszermagkód?

Amikor a rendszermagot valamilyen modul betöltésére utasítjuk, a ***kernel/module.c*** állományban megadott kód fut le. Itt szinte minden feladatot a `load_module` függvény hajt végre: a modult megfelelő szakaszokra tördeli, ellenőrzi a memóriahelyzeteket és a szimbólumokat, illetve elvégzi azokat a további feladatokat, amelyeket egyébként az összeszerkesztő program (linker) szokott. A folt ezt a programot módosítja a következő sorok beszúrásával:

```
if (module_check_sig(hdr, sechdrs, secstrings)) {
    err = -EPERM;
     goto free_hdr;
}
```

A `module_check_sig` nevű új függvény tartalmazza a modul-aláírási logikával kapcsolatos összes részletet. Ha hibát ad vissza, akkor ***Improper Permission*** hibaüzenetet adunk vissza a felhasználónak és megszakítjuk a modulbetöltést. Ha a függvény 0 értékkel tér vissza, azaz nem történt hiba, s a modulbetöltés folyamata sikeresen folytatódhat.
A `module_check_sig` függvény a ***kernel/module-sig.c*** állományba kerül. Ez a függvény először is ellenőrzi, hogy van-e aláírás a modulban – ezt a következő kódsorokkal végezzük el:

```
sig_index = 0;
for (i = 1; i < hdr->e_shnum; i++)
    if (strcmp(secstrings+sechdrs[i].sh_name,
               "module_sig") == 0) {
        sig_index = i;
        break;
}
if (sig_index <= 0)
    return -EPERM;
```

A kódrészlet a magmodul valamennyi ELF-szakaszán végiglépked, és kikeresi közülük a `module_sig` nevűt.
Ha az aláírást nem találja meg, hibát ad vissza és megakadályozza a modul betöltését. Ha megtalálja, folytatódhat a függvény végrehajtása.
Miután a rendszermag megtalálta a modul aláírását, meg kell állapítania, hogy milyen nyalábolóérték (hash value) tartozik a betöltendő modulhoz. Ehhez létrehozza a rendszermag által használt végrehajtható vagy adat-ELF szakaszok SHA1 nyaláboló értékét. A rendszermagban már megtalálható az SHA1 nyalábolók létrehozásához használt kód (egyéb nyalábolókkal, az MD5-tel és MD4-gyel együtt), így e lépés belső logikájának a nagy része már eleve készen van.
A függvény először az SHA1 algoritmus meghívásával foglalja le a titkosításátalakítási szerkezetet, majd ezt a következő kódsorokkal alaphelyzetbe állítja:

```
sha1_tfm = crypto_alloc_tfm("sha1", 0);
if (sha1_tfm == NULL)
    return -ENOMEM;
crypto_digest_init(sha1_tfm);
```

Az ELF-fájl számunkra érdekes részeinek SHA1 nyalábolóértékének előállítására az `sha1_tfm` változót használjuk, mint azt a *2. listán* látható kód mutatja.
A kódban csak azokkal az ELF-szakaszokkal foglalkozunk, amelyeknek a neve tartalmazza a „text” vagy a „data” szót, de amelyek nem foglalják magukban a „.rel.” szövegrészletet. Miután az összes szakaszt megtaláltuk és betöltöttük az SHA1 algoritmusba, a következő sorokkal az SHA1 nyalábolót áthelyezzük az `sha1_result` változóba:

```
crypto_digest_final(sha1_tfm, sha1_result);
crypto_free_tfm(sha1_tfm);
```

Az SHA1 nyaláboló kiszámításának és az aláírt nyaláboló helyének a megállapítása után már csak annyi a dolgunk, hogy visszafejtsük az aláírt nyalábolót és összehasonlítsuk a számított értékkel. Ezt a lépést a függvény utolsó sorával végezzük el:

```
return rsa_check_sig(sig, &sha1_result[0]);
```

Az `rsa_check_sig` függvény a ***security/rsa/rsa.c*** fájlban található és a rendszermagban futtathatóvá alakított eredeti GnuPG-kódot használja – ennek segítségével vissza lehet fejteni az aláírásokat és össze lehet hasonlítani az értékeket. Működésének pontos ismertetése azonban már meghaladná cikkünk kereteit.

Hogyan működik a felhasználói tér kódja?

Miután láttuk, hogy milyen módon dönti el a rendszermag, hogy a modul helyes aláírással rendelkezik-e, már csak az a kérdés, hogy miképpen tesszük azt az aláírást a modulba? Két, a felhasználói térben futó program – az `extract_pkey` és a `mod` –, valamint egy apró parancsfájl, a `sign` található a rendszermagfoltban (a ***security/rsa/userspace/*** könyvtárban). A két programot a könyvtárban található `Makefile` futtatásával fordíthatjuk le. Az `extract_pkey` felhasználásával a nyílt kulcsot helyezhetjük el a rendszermagban, a `mod` programot pedig a ***sign*** parancsfájl használja a magmodulok aláírására.
A modulok aláírásához először is létre kell hoznunk egy RSA-aláíró kulcsot – ezt a `gnupg` program segítségével tehetjük meg. Az RSA aláíró kulcs létrehozásához a `gpg`-nek adjuk meg a `--gen-key` kapcsolót, lásd a *3. listát*.
Sorra válaszolgassuk meg a feltett kérdéseket, és végül elkészül az RSA-kulcsunk. A kulcs használatához ugyanakkor el kell készítenünk annak titkosított változatát is. Ehhez ismét futtassuk le a `gpg`-t, és szerkesszük át az imént készített kulcsot, lásd a *4. lista* (a kulcsomat a listában található szövegben `testkey`-nek neveztem el).
Mivel mi új kulcsot szeretnénk felvenni, az `addkey` parancsot gépeljük be:

```
Command> addkey
Please select what kind of key you want:
```

```
$ gpg --gen-key
gpg (GnuPG) 1.2.1; Copyright (C) 2002 Free
Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to
redistribute it under certain conditions.
See the file COPYING for details.

Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection?

RSA kulcsot akarunk készíteni, így először az 5-
öst választjuk, majd 1024-et adjuk meg.

Your selection? 5
What keysize do you want? (1024)
Requested keysize is 1024 bits
```

```
$ gpg --edit-key testkey
gpg (GnuPG) 1.2.1; Copyright (C)
2002 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to
redistribute it under certain conditions.
See the file COPYING for details.

Secret key is available.

gpg: checking the trustdb
gpg: checking at depth 0
➥signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
pub  1024R/77540AE9  created:
➥2003-10-09 expires: never trust: u/u
(1). testkey

Command>
```

```
   (2) DSA (sign only)
   (3) ElGamal (encrypt only)
   (5) RSA (sign only)
   (6) RSA (encrypt only)
Your selection?
```

RSA-kulcsra van szükségünk, ezért válasszuk a hatost, és válaszoljunk a fennmaradó kérdésekre. Miután a kulcs elkészült, gépeljük be a `quit` parancsot:

```
Command> quit
Save changes? yes
```

```
#!/bin/bash
module=$1
key=$2

# emeljük ki a bennünket érdeklő részeket
./mod $module $module.out

# sha1 alkalmazása a szakaszokra
sha1sum $module.out | awk "{print \$1}" >
➥$module.sha1

# szakaszok titkosítása
gpg --no-greeting -e -o - -r $key $module.sha1 >
➥$module.crypt

# a titkosított adat hozzáfűzése a modulhoz
objcopy --add-section module_sig=$module.crypt
➥$module

# ideiglenes fájlok eltávolítása
rm $module.out $module.sha1 $module.crypt
```

Most már van egy kulcsunk, amivel aláírhatjuk a modulokat. A modulok aláírásához a `sign` nevű egyszerű parancsfájlt használjuk, lásd az *5. listát*.
A parancsfájl először is a `mod` programot lefuttatja a magmodulon. Ez a program az ELF-állományból kiemeli a bennünket érdeklő részeket, majd egy ideiglenes fájlba menti őket. A `mod` programot a későbbiek folyamán ismertetjük részletesebben.
Miután létrehoztuk a bennünket érdeklő részeket tartalmazó ELF-fájlt, az `sha1sum` programmal SHA1 nyalábolót készítünk az állományról. Ezt az SHA1 nyalábolót fogjuk azután a GPG segítségével titkosítani, átadjuk a kulcsot, majd a titkosított állományt ideiglenes fájlba mentjük.
A titkosított állományt `module-sig` néven új ELF-szakaszként hozzáadjuk az eredeti modulhoz, amit az `objcopy` programmal végzünk el. Ennyi az egész. A linuxos gépeken már eleve létező szokásos programok segítségével nem volt nehéz az SHA1 nyalábolót előállítani, titkosítani és az ELF-fájlba illeszteni.
A `mod` program is igen egyszerű. Kihasználja, hogy az `objdump` nevű, `binutils` programon alapuló ***libbfd*** könyvtár tudja, hogyan kell kezelni az ELF-állományokat és többféleképpen is módosítani tudja őket. Mivel a ***libbfd*** könyvtár valamennyi nehéz ELF-logikát megvalósítja, a `mod` programnak nincs más dolga, mint a következő kód segítségével végiglépdelni az ELF-állomány kívánt szakaszain, lásd a *6. listát*.
Miután a magmodulokat alá tudjuk írni, a rendszermag pedig képes ezeket az aláírásokat ellenőrizni, már csak annyi maradt hátra, hogy a sikeres visszafejtés érdekében nyílt kulcsunkat a rendszermagba töltsük. A Linux-rendszermag levelezőlistán elég sok szó esik mostanában arról, hogyan is kell helyesen kezelni a kulcsokat. A vita során több jó ajánlás is született, amelyeket a 2.7-es rendszermagsorozat fog felhasználni. Egyelőre azonban nem kell aggódnunk a kulcsok helyes és rugalmas felhasználása miatt, úgyhogy egyszerűen közvetlenül beforagain:

```
for (section = abfd->sections;
     section != NULL;
     section = section->next) {
    if (section->flags & SEC_HAS_CONTENTS) {
        if (bfd_section_size(abfd, section)
        ➥== 0)
            continue;

        /* Csak azokkal a szakaszokkal
           foglalkozunk, melyeknek
           "text" vagy "data" olvasható
            a nevükben*/
        name = section->name;
        if ((strstr(name, "text") == NULL) &&
            (strstr(name, "data") == NULL))
            continue;

        size = bfd_section_size(abfd, section));
        data = (bfd_byte *)malloc(size);

        bfd_get_section_contents(abfd, section,
                                 (PTR)data, 0, size);

        stop_offset = size / opb;

        for (addr_offset = 0;
             addr_offset < stop_offset;
             ++addr_offset) {
            fprintf(out, "%c",
            ➥data[addr_offset]);
        }
        free(data);
    }
}
```

díthatjuk.
Először is szükség lesz nyilvános kulcsunknak egy másolatára. Utasítsuk a GPG-t, hogy a kulcsot egy ***public_key*** nevű állományba csomagolja ki:

```
$ gpg --export -o public_key
```

A GPG nyílt kulcsok kezelésének megkönnyítésére néhány Ericsson-fejlesztő egy ***extract_pkey*** nevű egyszerű programot készített, ami segít részeire bontani a kulcsot. Egy kicsit módosítottam ezen a programon olyan módon, hogy C-kódot készítsen a nyílt kulcshoz.
Futtassuk az ***extract_pkey*** programot, megadva neki a korábban elkészített ***public_key*** állományt. A kimenetet mentsük az ***rsa_key.c*** nevű fájlba:

```
$ extract_pkey public_key > rsa_key.c
```

Ezután ezt az állományt helyezzük a ***security/rsa/*** könyvtár gyökerébe, lecserélve a saját nyilvános kulcsunkat:

```
$ mv rsa_key.c ~/linux/linux-2.6/security/rsa/
```

Létrehoztuk a szükséges RSA kulcspárt és a saját nyilvános kulcsunkat a rendszermagkönyvtárba helyeztük. Készítsük el a foltozott rendszermagot – ne felejtsük el bejelölni a ***Module signature checking*** lehetőséget –, majd telepítsük. Amint ezt a rendszermagot betöltjük, már csak a saját kulcsunkkal aláírt modulokat tudjuk betölteni, úgyhogy óvatosan járjunk el: kizárólag fejlesztői gépen használjuk a módszert.

Mi maradt még hátra?

Mint azt a cikkben bemutattuk, számos lépés szükséges a kulcs előállításához, a rendszermag aláírásához és a nyílt kulcs rendszermagba történő illesztéséhez. Ez ebben a formában egy meglehetősen csiszolatlan fejlesztői projekt. Amennyiben a rendszermagfejlesztők és általában a Linux-közösség számára elfogadhatóbbá szeretnénk tenni, a lépéseket önműködővé kell tenni. Egyszerűbb megoldásra van szükség az összes modul aláírásához és a nyílt kulcs kezeléséhez is. A képesség leegyszerűsítésének nyilvánvaló igénye mellett néhány egyéb elérendő célt is megfogalmazhatunk a projektben:

- Az RSA-kódot általános titkosító keretrendszerbe kell helyezni, hogy azt más rendszermagmodulok is használhassák.
- Lehetővé kell tenni, hogy a rendszermagban egyszerre több nyílt kulcs is létezhessen, illetve egyetlen gépen többféle forrásból származó modulok is futtathatóak legyenek.
- Érdemes egyszerűsíteni az aláírási logikát, hogy a GPG saját aláírási képességét, esetleg a `bsign` program megoldását használhassuk az egyedi `mod` program helyett.

Köszönetnyilvánítás

Szeretném megköszönni az Ericsson fejlesztőinek, hogy elkészítették a `digsig` nevű programot és rendszermagfoltot, ezzel téve lehetővé, hogy GPG-változatukat felhasználjam a rendszermaghoz. Korábban is készítettem ilyet, de a megvalósítás borzasztó lett; szerencsére kiadták a megoldásukat, ami nagy segítséget jelentett. A `digsig` rendszermagfolt segítségével a felhasználók programokat írhatnak alá, a rendszermag pedig semmilyen aláíratlan programot nem fog futtatni. A projektről további adatok a ➲ http://sourceforge.net/projects/disec címen érhetők el. Szeretnék továbbá köszönetet mondani alkalmazómnak, az IBM-nek, amiért lehetővé tette, hogy a projekten dolgozzak, és ***Don Marti***-nak, aki folyton piszkált, hogy készítsem el és fejezzem be ezt a cikket.

Linux Journal 2004. február, 117. szám

Greg Kroah-Hartman (greg@kroah.com)
Jelenleg a Linux-rendszerek USB és gyors csatlakoztatású (PCI Hot Plug) egységei rendszermagba épített meghajtóprogramjainak a fejlesztője. Az IBM-nél dolgozik és rendszermaggal kapcsolatos kérdésekkel foglalkozik.

Frissítés 2.6-os sorozatú rendszermagra

Mint azt a Programvadászatban is írtam, több dolgot is frissíteni kell, ha a rendszermagnak ezt az új vonalát akarjuk használni.
Az első ilyen nálam (Debian/GNU Linux Woody) a `modconf` program kényszerű frissítése volt. Bizonyára vannak olyanok, akik emlékeznek még a Potato hiányosságaira a 2.4-es sorozat kezelésében – úgy látszik, ez megismétlődik.
Miután a modullistát üresen találtam, a ***/etc/apt/sources.list*** fájlban kijavítottam a Woody (vagy ha úgy tetszik: a stable) bejegyzéseket, a SID-re kiadtam az `apt-get update`, `apt-get install modconf` parancsot. Máris csorgott a hálózatról a program új változata, a telepítés után pedig egycsapásra megszűnt az előbb említett gond.
A másik érdekesség számomra az eszközök elnevezése volt, ezek közül is az PS/2-es egeremé. Az X nem tudott elindulni, mert nem találta a „Core Pointer”-t, vagyis az egeremet, ugyanis eredendően a ***/dev/psaux*** volt a beállításfájlban megadva. Ezt ***/dev/input/mouse0***-ra kellett kijavítanom.
A frissítés során talán a legfurcsább helyzet akkor áll elő, ha előtte nem `initrd`-s rendszermagunk volt. (Az `initrd` egy a rendszerindításhoz használt memórialemez- (ramdisk) fájl.) Ha viszont a Debian alatt a gyökérkönyvtárban ***initrd.img*** fájlt találunk, akkor az azt jelenti, hogy valószínűleg az előző rendszermagunk is `initrd`-s volt.

A rendszerbetöltőnkhöz hozzá kell adnunk az alábbi sorokat, különös tekintettel az `initrd`-bejegyzésre:

GRUB:

```
title Debian2.6
          root (hd0,2)
          kernel /vmlinuz root=/dev/hda3
          initrd /initrd.img
```

LILO:

```
image=/vmlinuz
label=Linux
read-only
initrd=/initrd.img
```

Természetesen mindenki a saját rendszeréhez igazítva módosítsa a fenti beállításokat .
A felhasználók biztosan találkozni fognak ilyen, vagy ehhez hasonló dolgokkal, bármelyik Linux-változatot futtassák is. Ha előfordul olyan érdekesség, ami a többi felhasználó számára is hasznos lehet, kérek midenkit, hogy írja meg nekem a csontos.gyula@linuxvilag.hu címre.

Csontos Gyula

Létrehoztuk a szükséges RSA kulcspárt és a saját nyilvános kulcsunkat a rendszermagkönyvtárba helyeztük. Készítsük el a foltozott rendszermagot – ne felejtsük el bejelölni a ***Module signature checking*** lehetőséget –, majd telepítsük. Amint ezt a rendszermagot betöltjük, már csak a saját kulcsunkkal aláírt modulokat tudjuk betölteni, úgyhogy óvatosan járjunk el: kizárólag fejlesztői gépen használjuk a módszert.

Mi maradt még hátra?

Mint azt a cikkben bemutattuk, számos lépés szükséges a kulcs előállításához, a rendszermag aláírásához és a nyílt kulcs rendszermagba történő illesztéséhez. Ez ebben a formában egy meglehetősen csiszolatlan fejlesztői projekt. Amennyiben a rendszermagfejlesztők és általában a Linux-közösség számára elfogadhatóbbá szeretnénk tenni, a lépéseket önműködővé kell tenni. Egyszerűbb megoldásra van szükség az összes modul aláírásához és a nyílt kulcs kezeléséhez is. A képesség leegyszerűsítésének nyilvánvaló igénye mellett néhány egyéb elérendő célt is megfogalmazhatunk a projektben:

- Az RSA-kódot általános titkosító keretrendszerbe kell helyezni, hogy azt más rendszermagmodulok is használhassák.
- Lehetővé kell tenni, hogy a rendszermagban egyszerre több nyílt kulcs is létezhessen, illetve egyetlen gépen többféle forrásból származó modulok is futtathatóak legyenek.
- Érdemes egyszerűsíteni az aláírási logikát, hogy a GPG saját aláírási képességét, esetleg a `bsign` program megoldását használhassuk az egyedi `mod` program helyett.

Köszönetnyilvánítás

Szeretném megköszönni az Ericsson fejlesztőinek, hogy elkészítették a `digsig` nevű programot és rendszermagfoltot, ezzel téve lehetővé, hogy GPG-változatukat felhasználjam a rendszermaghoz. Korábban is készítettem ilyet, de a megvalósítás borzasztó lett; szerencsére kiadták a megoldásukat, ami nagy segítséget jelentett. A `digsig` rendszermagfolt segítségével a felhasználók programokat írhatnak alá, a rendszermag pedig semmilyen aláíratlan programot nem fog futtatni. A projektről további adatok a ➲ http://sourceforge.net/projects/disec címen érhetők el. Szeretnék továbbá köszönetet mondani alkalmazómnak, az IBM-nek, amiért lehetővé tette, hogy a projekten dolgozzak, és ***Don Marti***-nak, aki folyton piszkált, hogy készítsem el és fejezzem be ezt a cikket.

Linux Journal 2004. február, 117. szám

Greg Kroah-Hartman (greg@kroah.com)
Jelenleg a Linux-rendszerek USB és gyors csatlakoztatású (PCI Hot Plug) egységei rendszermagba épített meghajtóprogramjainak a fejlesztője. Az IBM-nél dolgozik és rendszermaggal kapcsolatos kérdésekkel foglalkozik.

Frissítés 2.6-os sorozatú rendszermagra

Mint azt a Programvadászatban is írtam, több dolgot is frissíteni kell, ha a rendszermagnak ezt az új vonalát akarjuk használni.
Az első ilyen nálam (Debian/GNU Linux Woody) a `modconf` program kényszerű frissítése volt. Bizonyára vannak olyanok, akik emlékeznek még a Potato hiányosságaira a 2.4-es sorozat kezelésében – úgy látszik, ez megismétlődik.
Miután a modullistát üresen találtam, a ***/etc/apt/sources.list*** fájlban kijavítottam a Woody (vagy ha úgy tetszik: a stable) bejegyzéseket, a SID-re kiadtam az `apt-get update`, `apt-get install modconf` parancsot. Máris csorgott a hálózatról a program új változata, a telepítés után pedig egycsapásra megszűnt az előbb említett gond.
A másik érdekesség számomra az eszközök elnevezése volt, ezek közül is az PS/2-es egeremé. Az X nem tudott elindulni, mert nem találta a „Core Pointer”-t, vagyis az egeremet, ugyanis eredendően a ***/dev/psaux*** volt a beállításfájlban megadva. Ezt ***/dev/input/mouse0***-ra kellett kijavítanom.
A frissítés során talán a legfurcsább helyzet akkor áll elő, ha előtte nem `initrd`-s rendszermagunk volt. (Az `initrd` egy a rendszerindításhoz használt memórialemez- (ramdisk) fájl.) Ha viszont a Debian alatt a gyökérkönyvtárban ***initrd.img*** fájlt találunk, akkor az azt jelenti, hogy valószínűleg az előző rendszermagunk is `initrd`-s volt.

A rendszerbetöltőnkhöz hozzá kell adnunk az alábbi sorokat, különös tekintettel az `initrd`-bejegyzésre:

GRUB:

```
title Debian2.6
          root (hd0,2)
          kernel /vmlinuz root=/dev/hda3
          initrd /initrd.img
```

LILO:

```
image=/vmlinuz
label=Linux
read-only
initrd=/initrd.img
```

Természetesen mindenki a saját rendszeréhez igazítva módosítsa a fenti beállításokat .
A felhasznalók biztosan találkozni fognak ilyen, vagy ehhez hasonló dolgokkal, bármelyik Linux-változatot futtassák is. Ha előfordul olyan érdekesség, ami a többi felhasználó számára is hasznos lehet, kérek midenkit, hogy írja meg nekem a csontos.gyula@linuxvilag.hu címre.

Csontos Gyula

Nyomtatókapu közvetlen programozása

A legtöbb számítógépben, legyen az hordozható vagy asztali gép, a mai napig megtalálható a párhuzamos vagy nyomtatókapu (port).

A Centronics kapu elnevezés burkoltan az alkalmazott protokollt is tartalmazza, de ez mára már nem teljesen igaz. Jelenleg a legtöbb nyomtató az IEEE 1284 szabvány ajánlásainak megfelelően működik, ami lehetővé teszi az önműködő felismerést, valamint az egyéb kétirányú kapcsolattartást.
Nos, mire is jó a nyomtatókapu? Önként adódik a válasz: nyomtatásra. Ez viszont ma egyre kevésbé igaz, ugyanis a legtöbb új nyomtató különböző újabb kapukat használ, leginkább az USB-t, de előfordul – különösen a hordozható változatoknál – az infra vagy a Bluetooth (rádiós) adatátvitel is. Korábban nyomtatókapus lapolvasók is léteztek, de ezek hamarabb álltak át az USB-kapura, mint maguk a nyomtatók.
Így viszont egy kissé haszontalannak tűnhet a nyomtatókapu, de ha már ott van, használjuk valamire! Régebben a nyomtatókapuhoz ethernet- (még régebben arcnet-) illesztők is megtalálhatók voltak, amelyek – ha némileg lassabban is, mint az „igazi" ethernet, de – használhatóak voltak. Ha viszont nincs ilyenünk, de két linuxos géppel is rendelkezünk és ezeket össze szeretnénk kötni, megtehetjük a dolgot magának a nyomtatókapunak a használatával is.

Hálózat a nyomtatókapun keresztül

Előfordulhat, hogy két linuxos gépet össze akarunk kötni, de nem azonos a bennük található hálózati kártyák hálózatkezelése, vagy csak nem akarjuk egy másik hálózatról átállítani, esetleg azt a másik hálózatot is el akarjuk érni. Megeshet, hogy a régebbi hordozható gépekben nincs is hálózati csatoló. Nos, ha van szabad nyomtatókapu a gépekben, megoldhatjuk a problémát. Mire van hozzá szükségünk?

- Mindkét gépben egy-egy szabad nyomtatókapura.
- Egy nullnyomtató kábelre, laplinkként is ismert, esetleg csak egy átalakítóra, s a nyomtatókábel végére.
- A gépeken telepített GNU/Linux-rendszerre (esetleg más Unixra).

Mit is kell tennünk?

1. Először csatlakoztatni kell a kábel két végét a gépekhez.
2. Ha az `lp` modult használjuk, akkor addig távolítsuk el, amíg a nyomtatókaput a hálózati feladatokhoz használjuk (`rmmod lp`).
3. Ha még nincs bent, töltsük be a `parport` modult (`insmod parport`).
4. Ezt követően a megfelelő értékekkel töltsük be a `parport_pc` modult (például `insmod parport_pc io=0x378 irq=7`).
5. A `plip` modult (`insmod plip`) is töltsük be.
6. A rendszerüzenetek között nézzük meg, hogy a plip kapunk (`dmesg`) milyen eszközszámot kapott, ez általában `plip0`.
7. A két gépen állítsuk be a csatolót (ez az IP-cím csupán szemléltetésül szolgál, a lényeg, hogy egymásra hivatkozzanak):
   `ifconfig plip0 192.168.10.1 ➥pointopoint 192.168.10.2 up`
   illetve
   `ifconfig plip0 192.168.10.2 ➥pointopoint 192.168.10.1 up`
8. A `ping` utasítás segítségével meggyőződhetünk arról, hogy a kapcsolat működik-e.
9. Ha az egyik gép a másikon keresztül más hálózatot is el akar érni, akkor ezen átjáróként állítsuk be a másik gépet:
   `route add default gw ➥192.168.10.1`
10. Továbbá a másik gépet állítsuk be útválasztónak (azaz engedélyezzük az IP-csomagtovábbítást).

A Linux nyomtatókapu-illesztője

Röviden tekintsük át, hogyan is használják a programok a párhuzamos kaput Linux-rendszer alatt. Előbb azonban ismét vessünk egy pillantást a múltba: a DOS időszakában a hozzáértők számára egyszerű volt a különböző kapuk és általában a gép alkatrészeinek az elérése – ezt a megfelelő kapucímek írásával és olvasásával lehetett megoldani. Például, ha a nyomtatókapu a 0x3bc címen helyezkedett el (hogy igazán régi legyen a példa, mert ezt a címet általában a mono-, Herkules és CGA kártyák beépített nyomtatókapuja használta), akkor egy 8 bites értéknek a nyomtatókapu adatvonalaira való kiírása a következőképpen történt:

```
outb(0x3bc,data);
```

vagy assemblerben:

```
mov     dx,0x3bx
out     dx,ax      ; ax=data
```

Nos, ez természetesen már a múlté… Mivel sokféle nyomtatókapu létezik (na jó, nem sok, de többféle, például a PC-s változat, a Macintoshé, a Sun munkaállomásé), valamint többféle csatlakoztatható eszköz van, célszerű volt egy illesztőfelületet írni, amit ***Tim Waugh*** mások segítségével meg is tett.

A felépítés lényege, hogy az eszközök elérése többrétegűvé vált (ez természetesen más elemekre is igaz, nem csak a nyomtatókapura). Legalul egy közvetlenül az adott vas típusától függő modul helyezkedik el, esetünkben a `parport_pc`. Ennek meg is adhatjuk a pillanatnyi jellemzőket, ha tudjuk – például a betöltése ilyesmi lehet:

```
parport_pc io=0x3bc irq=none
```

A következő modul a `parport`, amely egységes elérési felületet biztosít a különböző tényleges kapumegvalósításokhoz. Innentől kezdve a nyomtatókapu a gép megvalósításától függetlenül egységes módon érhető el. A `parport` programozási felületét használva működnek a „magasabb” szinten lévők: a leggyakoribb az `lp`, ez valóban nyomtatót illeszt a rendszerbe, de ilyen még az előbb látott `plip` is. A nyomtatókapu használata a `parport` modul függvényein keresztül lehetséges, azonban csak úgy, ha rendszermagmodult (drivert) készítünk. Ennek a modulnak a következő feladatai vannak:

- Be kell jegyeztetnie magát mint a nyomtatókapu felhasználóját (registration).
- Le kell kérdeznie, hogy milyen kapuk vannak a rendszerben (ezt visszahívó, azaz callback függvények segítségével valósítja meg: `attach` és `detach`).
- Le kell foglalnia a kaput (`claim`), ami blokkoló és nem blokkoló módon is történhet, azaz vagy vár, amíg szabad lesz a kapu, vagy nem, ekkor viszont nem biztos, hogy sikerült lefoglalni.
- Ezután a `parport` és `parport_dev` szerkezeteken keresztül használhatja a kaput.
- A kapu használatát `yield` típusú függvények segítségével ideiglenesen fel lehet adni.
- Végül a `release` függvény segítségével a használat jogát más folyamatoknak adhatja át.

A `parport` modul programozásáról, valamint egy „próba” `lp` meghajtó írásával kapcsolatban sokat megtudhatunk a Tim Waugh által írt parportbook dokumentumból vagy a Linux-mag forrásszövegéhez mellékelt leírásokból.

Közvetlen kapuelérés

Mint láthattuk, a kapu elérése meglehetősen bonyolult dolog. Ez nagy sorozatban gyártott eszközök esetén nem annyira nagy baj, mert előbb-utóbb úgyis megírja valaki a szükséges meghajtóprogramot (esetleg többen is, netán magunk is besegíthetünk). Ha azonban mi magunk akarjuk használni a kapukat a saját készülékünkkel való „beszélgetésre”, akkor nehezebb helyzetben vagyunk. Kedvenc vesszőparipám és munkám mellett hobbim is a mikrovezérlők programozása. Ezekhez szerettem volna egy programbetöltő készüléket (régebbi nevén „égetőt”) készíteni. Mindemellett elsőrendű célom az volt, hogy lehetőleg mindenféle rendszerből lehessen használni, azaz Linux, Windows és – horribile dictu! – DOS alól is. Ezért a nyomtatókapuhoz való csatlakoztatás mellett döntöttem. Ennek külön előnye, hogy még az újabb hordozható gépeken is megtalálható, így mindenfelé használható lesz.
A DOS-változattal a fent említettek szerint nincs gond, ugyanakkor a másik kettőnél jócskán kijutott belőle. Mire van szükség egy ilyen készülék illesztésénél? Digitális ki- és bemenetekre, mégpedig úgy, hogy mind beolvasni, mind kiírni bitenként lehessen őket – tehát nem elegendő, amit a nyomtatóillesztő programok nyújtanak, hogy tudniillik egész bájtokat küldünk a kapura és visszafelé csak a nyomtató állapotjeleit (illetve azok közül is csak néhányat) olvassuk. Lássuk legelőször azt, hogy mit is tud nyújtani a nyomtatókapu:

- egy 8 bites adatkaput, amely az újabb gépeken kétirányú, a régebbieken csak kimenet;
- egy 5 bites állapotkaput (status), amely bemenet;
- egy 4 bites vezérlő (control) kaput, amely nyitott kollektoros kimenet felhúzó ellenállással, és vissza is olvasható, így elvileg bemenetként szintén használható lenne, de ez nem minden gépen működik (esetenként más módon van megvalósítva).

A kapuk címei a báziscímhez (B, ami a fent említett 0x3bc mellett 0x278 és 0x378 szokott még lenni, az új BIOS-okban választhatóan) a következők:

- adatkapu B+0 (például: 0x278),
- állapotkapu B+1 (például: 0x279),
- vezérlőkapu B+2 (például: 0x27a).

Ezeket a címeket szeretnénk írásra és olvasásra elérni. Ha az `outb` és `inb` függvényekkel próbálkozunk, mint DOS-ban, akkor szakaszolási hibát (segmentation fault) kapunk, lévén, hogy az adott ki-bemeneti terület nem tartozik a programunkhoz. Tehát a magon (kernel) keresztül kell elérnünk. Szerencsére a `parport` modul szerzője `ioctl` hívásokon keresztül lehetővé tette a kapu közvetlen írását és olvasását. Ezt elég szegényes módon említik meg a leírásokban, és a neten is kevés ilyen jellegű útmutató található, ezért is osztom meg kutatásaim eredményét az érdeklődőkkel. Először is akciónk sikeréhez az kell, hogy a `parport` modul (ez pedig függ a `parport_pc` modultól) be legyen töltve. Itt említem meg, hogy a viszonylag újabb magok (2.4) esetében a `parport0` eszközfájl (***/dev/parport0***) független az adott kapu báziscímétől: mindig az elsőként beállított kapuhoz kapcsolódik, a `parport1` a másodikhoz stb. Régebben ezek a fizikai címekhez kötődtek.
Ezután arra van szükség, hogy a felhasználó, aki futtatni akarja a programot, írási és olvasási jogosultsággal bírjon a `parport0` eszközfájlon. Ha mindez megvan, akkor programunkban a következő fejlécfájlokat az alábbi módon be kell olvastatnunk:

```
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/io.h>
#include "ppdev.h"
```

Az utóbbi azért a pillanatnyi könyvtárból lesz beolvasva, mert ha nem rendszergazdaként és magfordítás közben dolgozunk, akkor nincs benne az ***include*** állományok elérési útjában (általában csak magfordításnál szükséges), ezért átmásoltam a forrásállományok könyvtárába. Tulajdonképpen ez az utóbbi tartalmazza a `parport` eszközhöz szükséges meghatározásokat. Régebbi rendszereken esetleg ***ppuser.h*** is lehet a neve, a használat szempontjából csaknem azonosak. Az adott fájlt a mag forrásfájában kell megkeresnünk.

Miután mindezt megtettük, a következő módon használhatjuk a kaput:

- `hLpt= open("/dev/parport0", O_RDWR);` – megnyitja az eszközfájlt.
- `ioctl(hLpt, PPEXCL,0);` – kizárólagosságot kér a kapura, a következő hívás előtt kell beállítani.
- `ioctl(hLpt,PPCLAIM,0);` – használati igényt „jelent be” a kapura. 0 visszaadott érték esetén megvan, egyébként nincs.
- `ioctl(hLpt,PPRELEASE,0);` – visszaadja a rendszernek a kapu használati jogát, utána a `close(hLpt)` hívással be lehet zárni az eszközfájlt.
- `ioctl(hLpt,PPRSTATUS,&data);` – az `unsigned char data` változóba másolja az állapotkapu bitjeinek az értékét.
- `ioctl(hLpt,PPRDATA,&data);` – az `unsigned char data` változóba másolja az adatkapu bitjeinek az értékét; ez csak kétirányú változat esetén használandó.
- `ioctl(hLpt,PPWDATA,&data);` – a `data` változó tartalmát az adatkapura írja. Amit ide kiírunk, az a következő kiírásig változatlan marad, azaz a megfelelő logikai értéke lesznek a csatlakozó érintkezőin. (Ez a következő vezérlőkapura is igaz). Így felhasználható a külső eszközünk vezérlésére.
- `ioctl(hLpt,PPWCONTROL,&data);` – a vezérlőkapura írja a `data` változó tartalmát.
- `ioctl(hLpt,PPRCONTROL,&data);` – beolvassa a vezérlőkapu bitjeit. Ez nem biztos, hogy ugyanaz, mint amit kiírtunk, mivel külső eszköz megváltoztathatja a logikai szinteket, valamint a lentebb említett módon bizonyos bitek invertálódnak.
- Emellett vannak még a kétirányú változatok adatirányát beállító, az ezekhez tartozó kibővített vezérlőregisztert, a FIFO tárolót, és a magasabbrendű protokollt segítő hívások, de ezek túlmutatnak cikkünk témáján.

Mint azt a korábbiakban megemlítettem, az állapot- és vezérlőkapunak van néhány furcsa sajátossága. Az állapotkapu csak 5 bites, ezek a bájt felső részén olvashatóak, azaz az alsó három nem valós érték. Ráadásul a legfelső bit invertált, azaz a csatlakozón lévő logikai szint ellentétét mutatja. (A miértekre nem tudok felelni, valószínűleg így jobban megfelelt az eredetileg kialakított protokollnak.) Tehát, ha a kapun lévő valódi adatot meg akarjuk kapni, akkor a `data=data ^ 0x80;` utasítással meg kell ezt a bitet fordítanunk (ez egy bináris `xor` művelet, ha a műveleti jel netán ismeretlennek tűnne).
A vezérlőkapunál is vannak érdekességek: ez csak 4 bites, de ezek a bájt alsó részén helyezkednek el. Ráadásul több is invertált. Így a következő módon kell kiírnunk az adatot, ha azt akarjuk, hogy az 1 állapotú bitek logikai 1 értéket jelentsenek a csatlakozón:

```
data=data ^ 0x0b;
```

A kiírást követően, hogy a felső bitek ne zavarjanak (mivel néhány esetben ezek közül valamelyik vezérli a kétirányú adatkapu irányát, esetleg a megszakítási jel előállítását is):

```
data=data & 0x0f;
```

Ezután már kiírhatjuk a kapott adatokat a fent látott `ioctl(hLpt, PPWCONTROL,&data);` hívással.
Mint látjuk, némi küzdelem árán elérhetjük, hogy kedvenc Linux-rendszerünkből is ugyanúgy kezeljük a párhuzamos kaput, mint régen, a „mindent szabad” DOS-os időszakban. Ez előnyös lehet, ha saját építésű vagy más okból a rendszer által nem támogatott készüléket akarunk működtetni. Ilyen esetben a nehézkes kipróbálás miatt elég munkás a további fejlesztést C-ben végezni; erre jelenthet megoldást a – talán egy későbbi cikkben ismertetésre kerülő – beágyazott forth rendszer használata, amelyből megfelelő ingyenes, nyílt forrású változatot Linuxra is találhatunk.

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

Személyi videófelvevő építése

Csatlakoztassuk műholdas lavórunkat egy személyi számítógép alapú felvevőhöz, amelynek segítségével időeltolást végezhetünk kedvenc műsorainkban, archív másolatokat készíthetünk és a beépülő modulok segítségével milliónyi egyéb feladatot is megvalósíthatunk!

Manapság a tévéműsorok felvételét és szerkesztését egyaránt digitálisan végzik – mi mégis analóg rendszereken, kábelen, műholdon vagy földi állomások adásain keresztül vesszük őket. Nem is tudom, akad-e még valami előnye ennek a megoldásnak. Szerencsére létezik egy másik lehetőség is, amely jobb képminőséget, Dolby Digital hangot és elektronikus műsorfüzetet (Electronic Program Guide, EPG) kínál. Hölgyeim és uraim, ismerkedjenek meg a DVB, a digitális műsorszórás (digital video broadcasting) világával!

A DVB egy MPEG-2 tömörítésű folyamot továbbít, legnagyobb sávszélessége 15 Mbit/s. A DVB-t sosem használták csupán videofolyam továbbítására, a hang és az egyéb szolgáltatásokhoz szükséges adatok küldésére külön sávokat alkalmaznak. Minden adat kisméretű csomagokban utazik – természetesen, ha egyes csomagok kimaradnak vagy sérülten érkeznek meg, akkor a képen és a hangban is lehetnek hibák. A műsor azonban a helyesen megérkező csomagok adatai alapján megy tovább, mintha mi sem történt volna, vagyis összehangolási kérdésekkel nem kell foglalkoznunk. Különösebb bajok a jel vételével sem lehetnek, az antennákat általában úgy méretezik, hogy az eső, a hó vagy a kisebb állatok ne zavarják a tévézést.

A DVB három változatban létezik: a DVB-S műholdas, a DVB-C kábeles, a DVB-T pedig felszíni sugárzású rendszert takar. Alapvetően mindhárom rendszer ugyanaz, különbséget a vevőegységekben találni. A DVB-T eléggé új megoldás, még nem nagyon terjedt el, a másik kettő azonban – főleg a műholdas – világszerte széles körben kedvelt.

Építkezzünk!

A nagyobb gyártók korszerű set-top boxai érdekes lehetőségeket kínálnak, például képesek felvételeket készíteni merevlemezre, illetve az MP3 formátumú fájlokat is lejátsszák. Komolyabb dolgokat azonban nem tudnak, például SVCD vagy MPEG-4 formátumban nem tudunk velük felvételeket menteni. Mivel a DVD-írók és a -lemezek ma már megfizethetők, kedvenc tévéműsoraink DVD-re írása egyáltalán nem megvalósíthatatlan ötlet. Európában a digitális személyi videófelvevők (PVR) ára sajnos 500 dollár (nagyjából 110 ezer forint) körül mozog, és a legtöbb modellben nincs merevlemez. Egy nagy tudású DVB-S kártya viszont akár 165 euróért is beszerezhető a webes boltokból.

Mit ér azonban az olcsó vas, ha nincs hozzá megfelelő alkalmazás? A VDR alkalmazás segítségével, valamint a saját választásunk szerinti Linux-változattal és DVB-kártyával saját nagy teljesítményű set-top boxot építhetünk. Az általam összeállított gépen lehetett tévézni, műsorokat rögzíteni és időeltolást végezni bennük, valamint különleges szolgáltatásokra is képes volt, például MP3/Ogg-fájlok lejátszására, az MPlayer által ismert formátumok megjelenítésére és a felvett anyagok MPEG-4, video-CD vagy DVD formátumban történő archiválására. Egy készen vásárolt set-top boxnak bizony keményen össze kell szednie magát, ha ilyen széles körben akar szolgáltatásokat nyújtani.

A vas kiválasztása

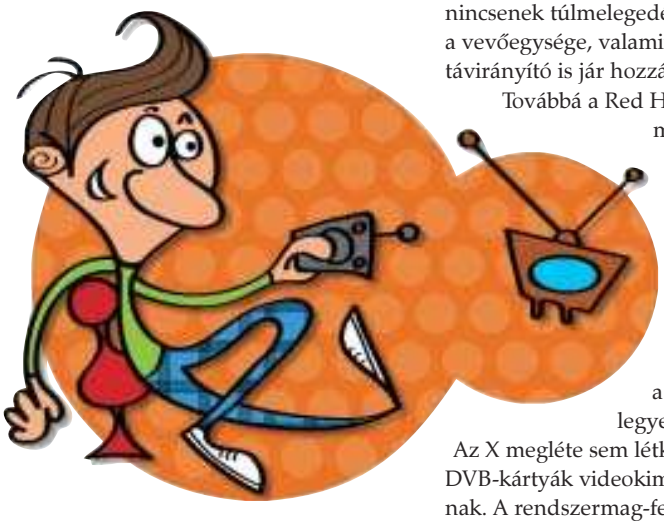
A gép összeállításához megfelelő alkatrészekre van szükség. Először is ne feledjük, hogy a felvételek rengeteg helyet foglalnak el. Egy 120 GB-os merevlemezre nagyjából 60 órányi videóanyag fér, ami átlagos esetben

1. kép A VDR főmenüje az Elchi folt telepítése után

2. kép A távirányító gombjainak megadása

bőven elegendő. Ha a filmeket gyakran archiváljuk és ilyenkor eltávolítjuk őket a merevlemezről, akkor akár egy kisebb merevlemezzel el is boldogulhatunk. Javaslom azonban, hogy 20 GB alá ekkor se menjünk – egy ekkora területen 3–4 film még kényelmesen elfér.

Gyors processzorra is szükség lesz. Ha a videóanyagokat tömöríteni, kódolni szeretnénk, akkor jó gyors kell, ha nem, akkor egy öregecske, 200 MHz órajelű is megteszi. Manapság újonnan



1700 MHz-es Celeronnál lassabbat nem kapunk, ami mindenféle feladatra bőségesen elegendő teljesítménnyel bír. A lejátszásokhoz használt MPlayer szintén gyors processzort igényel, legalább 1 GHz órajelűt; bár a pletykák szerint gyengébbel is működik. Én lassabb gépen is kipróbáltam az MPlayert és a képminőség bizony nem volt tökéletes. Mindennek oka abban a módszerben keresendő, ahogyan az MPlayer a DVB-kártya MPEG-2-dekóderét használja. A nem MPEG-1/2 formátumú anyagokat a program menet közben MPEG formátumúra hozza, ami leköt némi teljesítményt.

Ezzel el is érkeztünk a legfontosabb összetevőhöz, a DVB-kártyához: nagy tudású, beépített MPEG-dekóderrel rendelkező példányra van szükségünk. Az ilyen kártyák ugyan drágábbak, de számos hang- és tévécsatlakozóval rendelkeznek. A WinTV Nova és a hozzá hasonló kártyák inkább másodlagos választásnak jók, ha egyszerre több műsort is rögzíteni akarnánk. Ha tehetjük, a műholdas megoldást válasszuk; ez messze a legrugalmasabb rendszer, ugyanis semmilyen kábeles szolgáltatótól nem tesz bennünket függővé. Azt is megtehetjük, hogy több műholdas antennát állítunk üzembe, és így több csatorna vételi lehetőségét teremtjük meg. A továbbiakban a DVB-S változatról lesz szó, de a másik kettő beüzemelése is nagyon hasonló módon történik. Kártyaként én a Hauppauge Nexus-s típus mellett döntöttem, ami alighanem a legdrágább választás, viszont a régebbi modellekkel ellentétben nincsenek túlmelegedési gondjai, jó a vevőegysége, valamint infravörös távirányító is jár hozzá.

Továbbá a Red Hat Linux 9-es mellett kötöttem ki, de bármelyik változat megfelelő. Legyen a gépen GCC, fejlesztői csomagok és libjpeg, a többire ne is legyen gondunk. Az X megléte sem létkérdés, a jobb DVB-kártyák videokimenettel is bírnak. A rendszermag-fejlesztői csomagokat se felejtsük el telepíteni, a DVB-illesztőprogram fordításakor szükség lesz rájuk.

A meghajtóprogramok telepítése

Ha a kiválasztott terjesztés alapelemeit telepítettük, akkor fel kell tennünk a kártya meghajtóprogramját is. A CVS-változat a ➲ http://linvdr.org/download/vdr/Developer címen érhető el; írásom születésekor a ***linux-dvb.2003-09-05.tar.bz2*** a legfrissebb kiadás (az 57. CD Magazin/Video könytárában megtalálható a legfrissebb linux-dvb.2003-11-08.tar.bz2 változat). A legújabb meghajtóprogram időnként lefagy, ha a műholdas kábelt lehúzzuk vagy a vételi szint a nullára esik. Ilyenkor el kell távolítani és újra be kell tölteni az meghajtóprogramot, ami viszont nem mindig lehetséges – ebben az esetben nem marad más lehetőségünk, mint újraindítani a gépet. A lefagyások eléggé bosszantóak, ha éppen felvennénk valamit vagy pont egy film közepén járunk, de szerencsére elég ritkán keserítik meg az életünket.

Lépjünk be a ***/usr/src*** könyvtárba, csomagoljuk ki az meghajtóprogramot, majd a könyvtárát nevezzük át DVB-re. Az átnevezés fontos mozzanat, számos folt és beépülő modul ugyanis könyvtárnevek alapján működik. Lépjünk be a DVB könyvtárba és a `make` parancs kiadásával indítsuk el az meghajtóprogramot, valamint a hozzá társított – például a műholdak letapogatásával a csatornák listáját visszaadó – segédprogramok lefordítását. A `make install` parancsra nem lesz szükségünk, a modulok betöltését a ***runvdr*** parancsfájl végzi. A fordítás befejezését követően ne mulasszuk el a ***makedev.napi*** parancsfájlt lefuttatni – ez hozza létre a szükséges bejegyzéseket a ***/dev*** könyvtárban.

Csatornák keresése

Ha Európán kívül lakunk vagy nem az Astra műholdakat vesszük igénybe, akkor más lesz a csatornalistánk.

A csatornák keresése önműködően történik. A DVB-meghajtóprogramhoz tartozik egy scan nevű segédprogram, ezt a ***/apps/scan*** könyvtárban találjuk. A `-o vdr` kapcsolóval kell meghívni, ekkor a kimenet a VDR csatornaformátumában fog előállni. Ha a program által adott csatornalistát fájlba szeretnénk menteni, a szabványos kimenetet az alábbi paranccsal kell átirányítanunk:

```
./scan -o vdr > channels.conf
```

A VDR telepítése

Töltsük le a ***vdr-1.2.5.tar.bz2*** csomagot, bontsuk ki a ***/usr/src*** könyvtárba, majd lépjünk be. A VDR telepítése nem mindig egyszerű. Tudását különféle foltokkal feljavíthatjuk, csakhogy ha nem figyelünk eléggé, a sok foltozgatásba hamar belezavarodhatunk. Ha nagyszámú foltot szeretnénk feltenni, a legjobb az, ha egyesített folt formájában szerezzük be őket. Én az Elchi folton kívül mást nem használtam, ez a VDR csúnyácska alapértelmezett felhasználói felületét dobja fel egy kicsit. Ha figyelünk rá, hogy az adott VDR-változathoz készült foltot használjuk, nem lesz vele különösebb gondunk.

A beépülő modulok rengeteg szolgáltatást nyújtanak, az elektronikus levélben küldött értesítésektől kezdve egészen a „fullextrás” DVD-lejátszásig.

3. kép Az adatsorban a csatorna neve, az éppen futó műsor címe és a következő műsor címe látható

A következőkben két beépülő modul telepítéséről lesz szó: az egyik a távirányítót kezeli (remote), a másik az MP3/MPlayer-lejátszást teszi lehetővé. A `remote` beépülő modulra csak akkor van szükség, ha eredeti Hauppauge távirányítót használunk, az MP3/MPlayer ellenben hétköznapi halandó gépéről nem hiányozhat. Lépjünk be a */usr/src/vdr-1.2.5/PLUGINS/src* könyvtárba és bontsuk ki a két csomagot. A VDR-féle `Makefile` nem végzi el a beépülő modulok fordítását, amíg a könyvtárnevekből ki nem vesszük a változatszámot. A két könyvtárat tehát ***remote*** és ***MP3*** névre kell átkeresztelni.

Az mp3 beépülő modul működéséhez bizonyos további elemekre is szükség van, kicsit „kézzelfoghatóbban”: ez a ***libsndfile***, a ***libmad*** és a ***libid3tag***. Mivel a Red Hat-változatok MP3-támogatást nélkül készülnek, kézzel kell telepítenünk őket. Mindhárom összetevőt a ➲ http://www.freshrpms.org címről kiindulva szerezhetjük be.

Ha mindent beállítottunk, írjuk be a `make REMOTE=plugin NEWSTRUCT=1 all plugins` parancsot.

A `REMOTE=plugin` kapcsoló a használható beviteli módszerek körét a `remote` beépülő modullal bővíti. A Lirc segítségével kiválaszthatjuk, hogy milyen távirányítónk van. A videómagnókhoz készült példányokkal egészen jól boldogulhatunk, ilyenkor a `REMOTE=lirc` kapcsolót kell alkalmaznunk. A billentyűzet támogatása alapállapotban engedélyezve van, és jobb, ha ezen nem is módosítunk – hibakereséskor még jól jöhet.

A `NEWSTRUCT=1` kapcsoló feladata az, hogy a beépülő modulokkal közölje, hogy az új meghajtóprogramokat a */usr/src/DVB* könyvtárban kell keresniük. Mindent lefordítottunk, nincs más hátra, mint az indítóparancsfájl átírása. Alapként használjuk a `remote` beépülő modulhoz mellékelt példányt. A neve ***runvdr.remote***, a ***misc*** könyvtárban találjuk.

A parancsfájl egy billentyűtérképet tölt be, ezzel végzi a távirányítóról érkező jelek dekódolását. Minderre csak a Hauppauge infravörös vevőegységének használatakor van szükség. Ha nincs ilyenünk, akkor a parancsfájl szerkesztésekor a VDR gyökérkönyvtárában található ***runvdr*** parancsfájlból induljunk ki. Mozgassuk át a ***runvdr.remote*** parancsfájlt a VDR gyökérkönyvtárába, majd indítsuk el a kedvenc szövegszerkesztőnket. A 24. sorban találjuk a `vdr` indítását szabályozó kapcsolókat, amelyek nálam így néztek ki:

```
VDRCMD="$VDRPRG -w 60 -P scanner
        ➥-P\"mplayer -M
        ➥/video/plugins/
        ➥mplayer.sh\"
        ➥-P mp3 -P\"remote -i
➥/dev/input/event1\" $"
```

Ha bizonytalanok vagyunk abban, hogy mit kellene beleírnunk a fájlba, adjuk ki a `vdr -help` parancsot, amely az összes használható modult és a hozzájuk tartozó kapcsolókat is megjeleníti, köztük a VDR kapcsolóit is. Aggodalomra semmi ok, könnyen meg fogjuk találni, mivel kell bővítenünk a fájlt. Szükségünk lesz még egy alapkönyvtárra is, ennek alapértelmezett helyét a ***Make.config*** adja meg, a neve */video*. Minden beállításfájl és rögzített képanyag ebbe a könyvtárba kerül. A ***sources.conf*** fájlt és saját ***channels.conf*** fájlunkat másoljuk a */video* könyvtárba.

Az egyetlen dolog, ami még várat magára, az MPlayer/MP3 beépülő modul beállításfájljainak az elkészítése. Kezdjük azzal, hogy létrehozzuk a */video/plugins* könyvtárat. Két fájlra lesz szükségünk, az egyik az ***mp3sources.conf***, a másik az ***mplayersources.conf***. Nem kell félni, különösebben egyikkel sem fogunk megszenvedni. A kezdők az ***mp3sources.conf*** fájlba annyit írjanak, hogy `/video/music;Local files;0`, az ***mplayersources.conf*** fájlba gépeljék be a `/video/compressed;Local files;0` karakterláncot, majd mind a két fájlt mentsék. Ahhoz, hogy az MPlayer a DVB-kártyával együtt tudjon működni, újra kell fordítani. Töltsük le a forrást a ➲ http://www.mplayerhq.hu oldalról, majd a `--with-extraincdir=/usr/src/DVB/include` kapcsolóval gondoskodjunk a saját beállításaink érvényre juttatásáról. Futtassuk le a ***configure*** parancsfájlt, fordítsuk újra a programot, telepítsük – és elvileg végeztünk is.

Amint a ***runvdr*** parancsfájlból is kiderül, az MPlayer beépülő modul egy különleges héjparancsfájl, az ***mplayer.sh*** segítségével indítja az MPlayert; ezt a ➲ http://batleth.sapienti-sat.org/projects/VDR címről szerezhetjük be. A csomagban mindössze két fájl található, maga az ***mplayer.sh*** és az ***mplayer.sh.conf***, az utóbbi néhány beállítási értéket tárol. Ha a gépünk az MPlayer alapú lejátszáshoz túl lassú, akkor az ebben a fájlban található beállítások módosításával próbálhatunk meg javulást elérni.

KAPCSOLÓDÓ CÍMEK

Könyvtárak
➲ http://www.freshrpms.org

LIRC
➲ http://www.lirc.org

MPlayer
➲ http://www.mplayer.hu

mplayer.sh
➲ http://batleth.sapienti-sat.org/projects/VDR

Beépülő modulok, parancsfájlok és foltok a VDR-hez
➲ http://www.vdrportal.de/board/portal_downloads.php?site=6

Lassan kép is lesz?

Lépjünk vissza a */usr/src/vdr-1.2.5* könyvtárba és futtassuk a *runvdr.remote* fájlt. Ha Red Hatet használunk, akkor végezzük el az `LD_ASSUME_KERNEL=2.4.1` környezeti változó értékadását, a VDR ugyanis egyelőre nem működik a Red Hat által a legújabb változattal bevezetett natív Posix-réteggel. Sor kerül a DVB-kártya működéséhez szükséges modulok betöltésére, majd a VDR elindul. Kapcsoljuk be a tévét: fekete képernyőt kell látnunk, amely a távirányító gombjainak megadását kínálja fel. A varázsló futtatásának befejezése után készen állunk arra, hogy tévézzünk, felvételeket készítsünk, eltávolítsuk a reklámokat, MP3-at hallgassunk vagy videófilmeket nézzünk. A VDR gyökérkönyvtárában található egy kézikönyv, amely a műsorok rögzítését, szerkesztését és az időeltolás-szolgáltatás használatát taglalja.

Archiválás

Senki ne búslakodjon amiatt, hogy lassan végére ér a cikknek – bőven van még mit tenni, ha győzzük kedvvel és fáradtsággal. Mindjárt nézzük is meg például az önműködő archiváló szolgáltatást – sajnos akad néhány korlátja. Az (S)VCD mentés még gond nélkül működik, DivX alapú tömörítésnél viszont a program nem távolítja el a kép szélén található fekete csíkokat. Mindez eléggé hátrányosan befolyásolja a bitszámot, a méretet és az összesített képminőséget. Ha valóban jó minőségű, kisméretű MPEG-4-állományokat szeretnénk kapni, az archiválást kézzel kell elvégeznünk. A jelentősen javuló képminőség bőven kárpótol a fáradozásért.

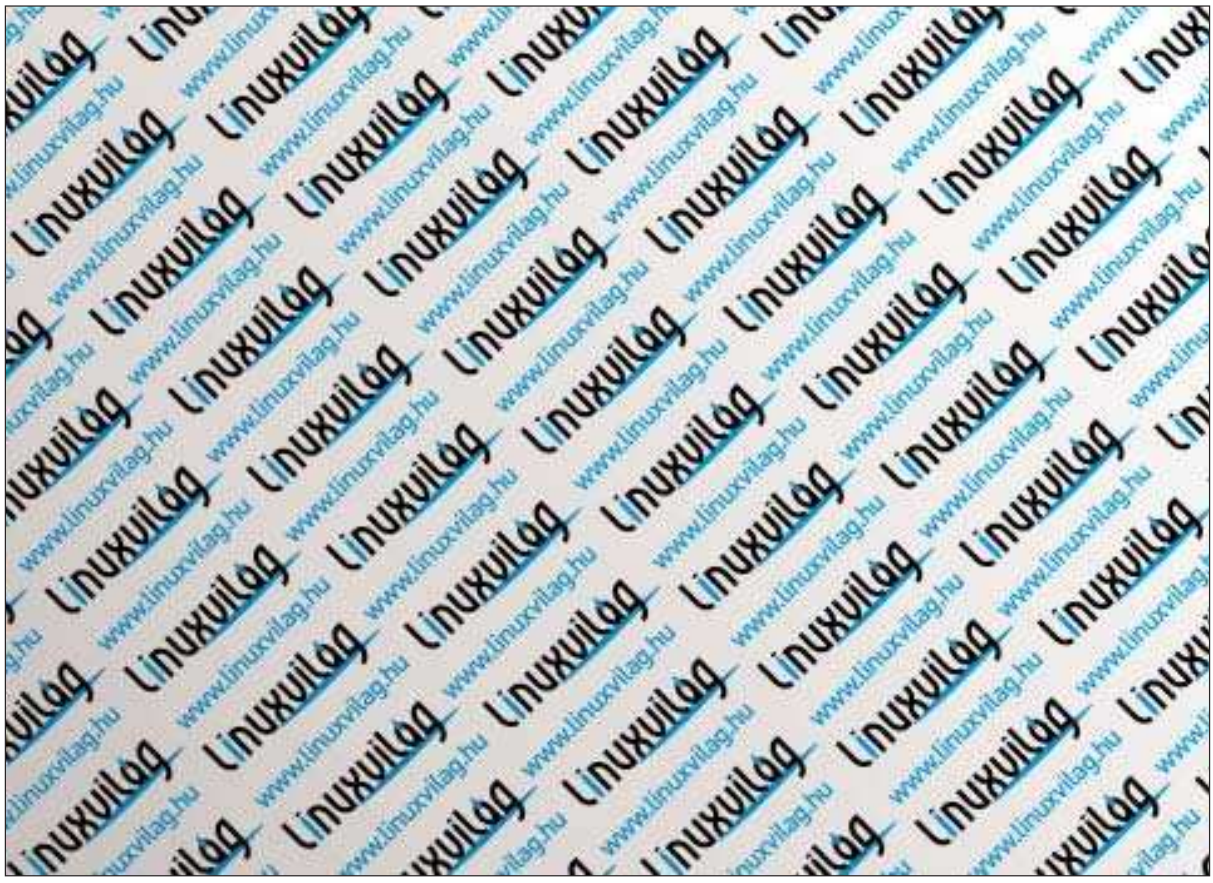
A VDR a felvételeket 2 GB-os fájlokra osztja, ami a mozgóképek átkódolásakor finoman szólva is: kényelmetlen. Ha az átalakítást kézzel végezzük, márpedig érdemes így eljárnunk, hiszen pontosabban tudjuk szabályozni a minőség/méret mutatót, a `mencoder` vagy a `transcode` megfelelő választást jelent. A `mencoder` gyors, MPEG-4 formátumú mentésekhez tökéletesen megfelel, a `transcode` viszont bőséges körítéssel bír. Ha nem akarunk sokat vacakolni, a VDRCONVERT a mi barátunk.

A ***README*** fájlban egyszerű telepítési leírást is találunk hozzá. Amíg letöltjük és lefordítjuk a programot, akár tévézhetünk is egyet. Ha lakhelyünkön NTSC szabványt használnak PAL helyett, akkor a VDRCONVERT-hez tartozó parancs- és beállításfájlok tartalmát módosítanunk kell, hogy a DVD/(S)VCD anyagok felbontása is tükrözze a szabvány előírásait.

Az egyetlen nagy baj az, hogy hiába van egy kiváló linuxos személyi videófelvevőnk, a tévéműsorok nem lesznek jobbak. Sajnos ebbe kénytelenek leszünk belenyugodni – minden nem megy egyszerre, nem igaz?

Linux Journal 2004. január, 114. szám

Christian A. Herzog
(noeffred@gmx.net)
Webes és nyílt forrású megoldásokkal foglalkozó programozó.

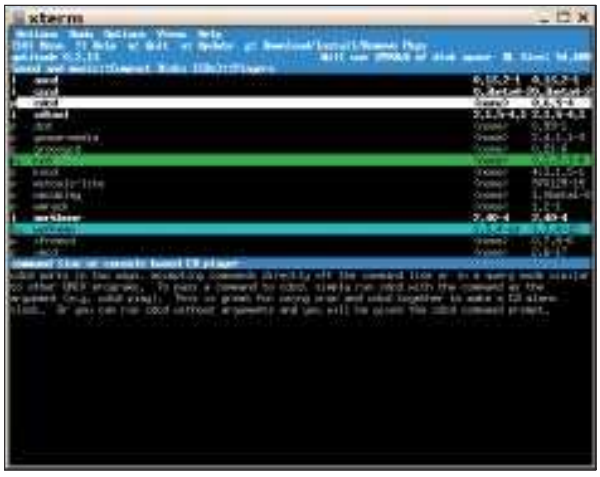


Debian otthon (4. rész)

A múlt hónapban a különböző hangrendszerekről szóltam, de igazán csupán addig jutottunk el, hogy valamiféle nyikorgást tudtunk előcsalni a gépből.

Ahogy írtam, a számomra legkényelmesebb módszer a gép által támogatott hangrendszerek kipróbálására az `xmms` használata. Nem kell más, csupán valamilyen hangfájl, majd a ***Beállítások*** ablakban (CTRL+P) kiválaszthatom a kívánt kimeneti modult (ALSA, OSS, ESD, vagy DiskWriter). A DiskWriter kimenet, ahogy azt a neve sugallja, nem megszólaltatja a lejátszott számot, hanem nyers WAV formátumban a lemezre írja.

De honnan kapar össze az ember zenét kipróbálási célzattal? A legkézenfekvőbb a hanglemez használata. Mindenki rendelkezik hivatalos hanglemezekkel, még az is, akinek házilag írott lemezei vannak. Mert ugyebár elméletileg a kis Artisjus-címkékért azért fizetünk, hogy jogosan írhassunk mindenféle zenét lemezeinkre. Ha rendszeresen hallgatsz hanglemezeket, előbb-utóbb keresel majd valamilyen kényelmes megoldást, ami egyszerű és az általad igényelt szolgáltatásokat nyújta. Például legyen képes lekérdezni a lemez tartalomjegyzékét az internetről. Mert mire jó az informatikai szuperhipersztráda, ha nem a kényelmi szolgáltatásokra? Létezik egy szolgáltatás, a CDDB, amelynek lényege, hogy lelkes felhasználók begépelgetik a lemezeik tartalomjegyzékét, majd amikor bárki a világon egy ugyanolyan szerkezetű lemezt kezd el hallgatni, a rendszer a már begépelt tartalomjegyzéket magától letölti.



1. kép Az aptitude

2. kép http://www.shoutcast.com

Tényleg csak érintőlegesen, de azért nézzünk bele a hanglemezlejátszók listájába!

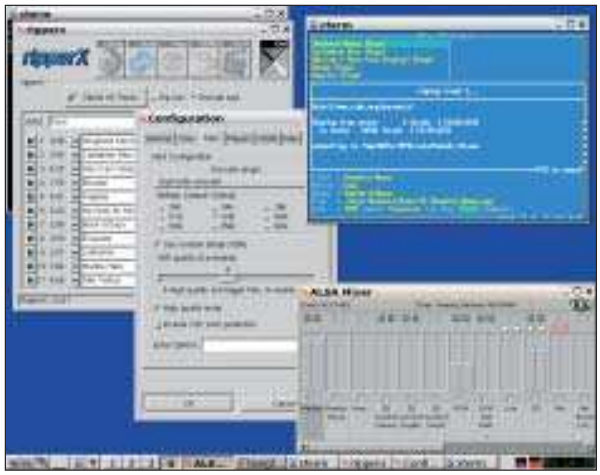
A két nagy környezet, a Gnome és a KDE alaplejátszói pont arra valók, amire szólnak: az adott környezethez legyen egy lejátszó. Most inkább nézzünk körbe a többi között. Kedvenc `aptitude`-unkban egy kategória szerinti lista-ablakban (F10, Views, New Categorical Browser) a ***Sound and Music :: Compact Disks :: Players*** alatt keressünk.

Az itt található programok közül a workman–workbone párost történelmi okokból érdemes kipróbálni. A workbone egy karakteres lejátszó, a workman pedig ennek egy grafikus felülettel ellátott változata. Mindenki válassza ki a neki megfelelőt. A saját kedvencem a `cccd`.

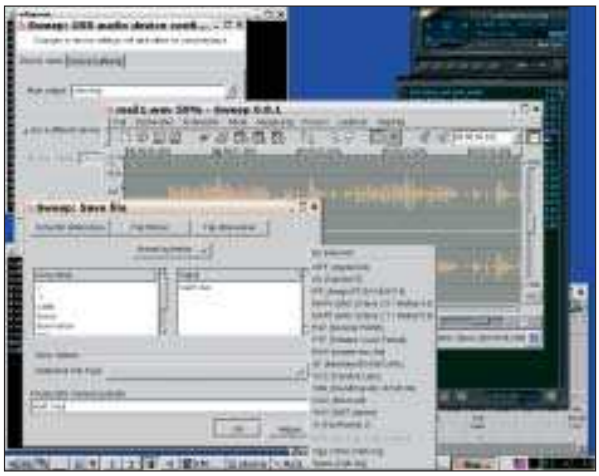
Jó, jó, ezzel kipróbáltuk a ***CD*** nevű csatornát, de mivel próbáljuk ki a ***PCM***-et (a számítógép által létrehozott hangot)? Ehhez elég egy wav vagy MP3, vagy bármilyen egyéb, lejátszható szám. De honnan szedjen ilyet a tévelygő felhasználó? Például a hanglemezből is készíthetünk wav-fájlokat – erre a legtöbb lemezégető program lehetőséget ad.

Lustább felhasználók kipróbálhatnak egy célprogramot is, mint amilyen a `grip` (ehhez Gnome kell), az `mp3c` (egy karakteres program) vagy a `ripperx`. Ahogy írtam, az `xmms` is rendelkezik egy lemezre író kimeneti modullal, így például netes rádiókból is fel tudunk számokat venni. Rengeteg netes rádiót találhatunk a ➲http://www.shoutcast.com címen. Remek! Ennyi harci előképzettséggel műsorokat vehetsz fel

netes rádiókból, majd kedvenc formátumodban tárolhatod őket vagy hanglemezt készíthetsz belőle (ehhez az `xcdroast`-ot ajánlom – a témára egy későbbi részben még bővebben kitérünk). Ha már a hangfájloknál tartunk, akkor egy hangszerkesztő programot is érdemes ajánlanom. Rengeteg olyan wav-szerkesztő van, amelyet csak azért érdemes letölteni, hogy utána gyakorolhassuk a ***csomag eltávolítása*** lehetőséget. Van például olyan hangszerkesztő, amelyik úgy kezdi, hogy az egész wav-fájlt benyalja, elemzi, készít egy kivonatot, azon dolgozhatsz, majd a végén az eredményt lemezre írja.
Remek! Elég, hogy felvegyél egy órányi anyagot a rádióból, amelynek eredménye egy 600 megás wav-fájl (nyers wav esetén nagyjából 10 megát számolhatunk egy percre), és egyből találkozol-e módszer nehézségeivel. Részemről a `sweep`-et ajánlom: könnyen és gyorsan használható, sőt szerkesztés közben még el is szórakozhatsz a nyikorgó hangokkal, amiket a fájlban mozogva hallhatsz. Mellesleg a `sweep` rendkívül sok átalakítást ajánl fel – ez a téma önmagában is nyugodtan megérdemelne néhány cikket. Itt jegyzem meg, hogy a linuxos programokra oly jellemző módon erre is igaz, hogy készítőik a hatásokat (effects) többnyire úgy írták meg, hogy több szerkesztőprogramban is használhatók. Nem arról van tehát szó, hogy minden cég készít egy hangszerkesztőt néhány átalakítóval vagy hatással, és ha egy épeszű darabot akarsz összerakni, akkor négy-öt programot kell használnod (mind fizetős, és nos, valljuk be, lehet, hogy az egyiket hivatalosan is megvásároltad, de nem túl valószínű); hanem a programozóknak az a céljuk, hogy az általuk készített modulokat a felhasználók tényleg használni tudják.
Pár bekezdés erejéig hadd térjek ki az MP3-mizériára. Ahogy írtam is: ha az ember nyers wav-fájlban, tömörítés nélkül kíván hanganyagot tárolni, átlagos, kétsávos (sztereó) módban, akkor percenként tíz megabájtot számolhat. Nagyjából így jön ki, hogy egy lemezre egy kicsit több mint egy órányi anyag fér fel (640 MB). Valójában a folyamatos hanganyagban nagyon sok zaj szerepel, amit az ember akár nem is hall, vagy ha hallja is, igazából csak bosszantja. Ha ezeket a zajokat kiszűrjük, a maradékot pedig (akár egy kevéssé gépigényes tömörítéssel) tömörítjük, akkor az eredmény egy szinte azonos minőségű fájl, de akár tizedakkora mérettel. Ilyen elmélet alapján született meg az MP3 (teljes nevén MPeg Audio Layer 3) tömörítés is. A hangot a tömörítések



3. kép A ripperx



4. kép A sweep

általában ***folyamként*** kezelik, azaz olyan adattömegként, amennyiből egységnyi idő alatt valamennyi adatnak elő kell állnia. Így célszerűen az ilyen folyamok egyik fontos mutatója, hogy milyen sávszélesség szükséges a hallgatásukhoz. Az MP3-as tömörítéssel például 64 bites szélesség mellett (ez nagyjából egy hétköznapi modem csúcssebessége) már élvezhető sztereozenét lehet létrehozni. A 192-es már rendkívül jó minőségnek számít, a 256-os pedig szinte fehér holló. Ez az adatszélesség annyit jelent, hogy egy egységnyi hanganyagot annyira kell tömöríteni, hogy „átférjen" egy ilyen átviteli vonalon.
A dolog remekül is működik, de sajnos az MP3 tömörítés jogvédett. A jog birtokosa pedig szemfülesen nem kér jogdíjat minden MP3 formátumban tárolt fájl után, sőt a lejátszóprogramok után sem, csupán az MP3-akat előállító kódolók után. Ezért ha az emberfia MP3-fájlokat akar készíteni, akkor vagy vásárol egy hivatalos kódolót, vagy jogsértő tevékenységet végez. Ez a szabad programok követői számára természetesen nem volt elfogadható, így nekiálltak, hogy készítsenek egy szabadon használható kódolót, ami (ha már lúd, legyen kövér) tömörítsen jobban, szebben. Így készült az Ogg-Vorbis páros, ezt a formátumot lényegében ma már minden komolyabb lejátszó támogatja. Linux alatt olyannyira, hogy az „MP3-készítők" alapértelmezett formátuma az Ogg. Ha tehát van véletlenül MP3-kódoló a gépünkön, használhatjuk, viszont nyugodtan átállhatunk Ogg formátumra. Ráadásul, ha birtokában vagyunk a megfelelő kódolóknak, olyan formátumba alakítjuk zenéinket, amilyenbe akarjuk.
Röviden végignéztük a hangkezelést, remélem, mindenki elboldogul ezen a területen, és bármikor el tudja végezni a szükséges alapfeladatokat, össze tud állítani egy neki tetsző hanglemezt, vagy éppen MP3-gyűjteményét tudja könnyedén gazdagítani, illetve mit is mondok! Természetesen Ogg-gyűjteményre gondoltam! Kellemes zenehallgatást!

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

A magyar Linux: UHU

A 2003. év nagy változást hozott a honi munkafelületek (desktop) és az irodai (office) rendszerek piacán: megjelent a magyar fejlesztésű és nyelvű UHU Linux csomag, ami a hazai informatika több részterületén is áttörést jelentett.

Eddigi előretörése egyenletes és folyamatos, sőt az idei évet úgy nyitotta, hogy a Sulinet Expressz jóvoltából döbbenetes sebességgel ismertté vált. Hazánkban gyakorlatilag a második helyen áll a Linux-változatok gyakorisági listáján, és komolyan veszélyezteti a SuSE Linux vezető helyét. Ezért úgy gondoltam, ideje ezt a magyar fejlesztésű rendszert alaposabb vizsgálatnak alávetni, hiszen elterjedtsége és támogatottsága folyamatosan nő. Most induló cikksorozatunkban a felhasználók számára a leghasznosabb és legérthetőbb formában igyekszünk kivesézni mindazt, amit az UHU Linuxról tudni érdemes. Sorozatunk első és második részében az általánosabb részekre térünk ki, ez a telepítési folyamatot, illetve a telepített és már működő rendszerünkben történő első felfedezőutunkat öleli fel. A későbbiekben a következő főbb témaköröket tekintjük át: a multimédiát, az internetet és a hálózatokat, a grafikát, az irodai megoldásokat, és bizony, bizony, a játékokat is, mivel a Linux a szórakozásnak ezt a formáját is támogatja.

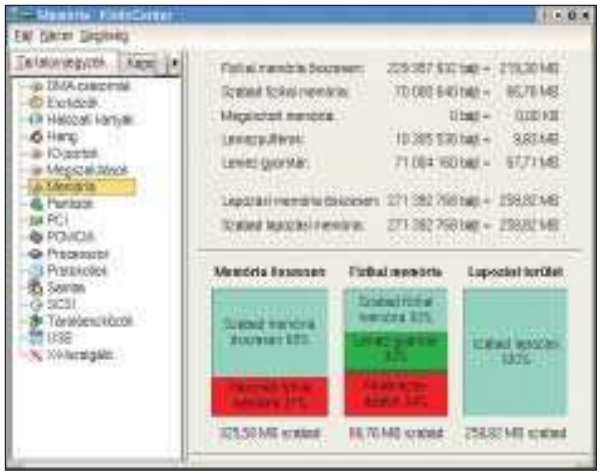
Az UHU Linux származása és alapjai

Mielőtt telepítenénk a rendszert, érdemes megismerni néhány apróságot. Többek között azt is, hogy honnan származik az UHU. A Linux-változatok két fő ágra oszthatók. A Debian Linux alapú rendszerekre, továbbá a Red Hat által bevezetett és azóta már de facto szabvánnyá vált Red Hat alapú megoldásokra. Jellemzően (származási ágtól függően) mindkét megoldás egy-egy csomagkezelési elvre épül, ez pedig a ***.rpm*** (Red Hat Package Manager), illetve a ***.deb*** (Debian). A linuxosok egymás között mindezt tömören csak rpm alapú vagy deb alapú rendszereknek hívják. Az UHU Linux a hagyományosnak tekinthető ***.deb*** elvet követi azzal a különbséggel, hogy egy kicsit javítottak rajta és megalkották a „dpkg" elvet, amelynek jellemző kiterjesztése a ***.uhu*** lehet. Ez azzal a veszéllyel járt együtt, hogy a ***.uhu*** csomagok szerényebb száma kedvezőtlen hatással lehet a rendszer elterjedésére, mivel ehhez külön ***.uhu*** csomagokat kell már meglévő ***.rpm***, illetve ***.deb*** csomagokból előállítani. Nos, nagyszámú önkéntes gondoskodott róla, hogy ez a nehézség ne okozzon fennakadást, és az UHU Linux-csapat vállalta, hogy ezeket az önkénteseket folyamatosan megtanítják ***.uhu*** csomagokat készíteni. Mint a későbbiekben látni fogjuk, a feladatot briliánsan megoldották, így igen nagyszámú – és a külön eljárásból következően – megbízható, ellenőrzött, és biztonságosan használható csomagok árasztották el honlapok tucatjait.

Gépigény

Az UHU Linuxszal kapcsolatos kérdések között nagyon sokszor szerepel, hogy „mégis, mekkora a gépigénye?". Erre maga a válasz is egy kérdés: milyen feladatra szeretnénk használni? A közhiedelemmel ellentétben ugyanis egy linuxos munkaállomás is használható játékgépnek. Kifogástalanul fut rajta a Quake, az Unreal, a Freespace, a Wolfenstein-sorozat összes jeles tagja, stratégiai játékok garmadája (Heroes3, MythII stb.), valamint a kisebb játékok százai. Értelemszerűen azonban ezeknek a gépigénye lényegesen nagyobb, sőt a többszöröse lehet például egy szövegszerkesztő erőforrásigényének. Ha pontosítom a választ, akkor azt lehet mondani, hogy a processzor legalább 300 MHz-es Celeron legyen, és habár 64 MB memórián is jól használható, érdemes 128 MB-nyit

alátenni. A teljes telepítés (amelynek során az első CD-ről mindent felteszünk – célszerű is így cselekedni) 1,9 GB helyet foglal el a merevlemezről, és ehhez természetesen hozzá kell tenni a feltétlenül szükséges csereterületet (swap), amelyet külön lemezrészre (partition) kell helyeznünk. Ezzel a gépkiépítéssel az UHU már olyan módon telepíthető, hogy zenelejátszásra és irodai műveletek elvégzésére egyaránt alkalmas, teljesen emberi sebességgel. Értelemszerűen a több erőforrás jobb, játékok esetében pedig egyenesen követelmény ennek a kiépítésnek akár a többszöröse is. (Nem kell ecsetelnem, hogy az



Unreal Tournament 2003 erőforrásigénye mekkora, hiába fut sokkal jobban Linuxon, mint más rendszereken.)
Érdemes azt is megjegyezni, hogy az UHU Linux nem „eszi" az erőforrást, ugyanakkor jellemzően követi a linuxos elvet, amely szerint az üresen hagyott erőforrás rossz erőforrás. Mindez az eszközök lehető legjobb kihasználása végett van így, de létezik olyan, manapság már nem meglepő méretű erőforrással rendelkező kiépítés, amely már nem képes a lehetőségeket teljesen kiaknázni, hiszen meghaladja az UHU teljes erőforrásigényét, akár több feladat egyidejű elvégzése esetén is. Lássunk egy példát: az én gépem egy 800 MHz-es Duron processzort tartalmaz 512 MB DDR memória kíséretében. Ezzel a kiépítéssel csereterületet szinte nem is használ, még a játékok alatt sem. Gyakorlatilag sokkal több erőforrás áll rendelkezésre, mint amennyi szükséges, ezért ezen a gépen bámulatosan gyorsan fut az UHU.

A telepítés fontosabb állomásai

A telepítése gyakorlatilag gyerekjáték. Nem térek ki a teljes telepítésre, aminek több oka is van: először is rendkívül beszédes a telepítő, teljesen egyértelmű, hogy mit miért kell tennünk. Ha egy külön merevlemezt szánunk az UHU-nak, szinte nem is kell más gombot megnyomni, mint az ***Ok***-t.
A másik ok, amiért nem részletezem a folyamat állomásait, az az, hogy a második korongon a teljes UHU Linux-füzet PDF formátumban megtalálható, s ez több fejezetnyi szöveget és képet tartalmaz ezzel kapcsolatban. Ennek a füzetnek (és a könyvnek, mivel az még tartalmasabb) a frissített változatai megtalálhatóak az ➲ ftp.uhulinux.hu kiszolgálón, s innen könnyedén letölthetők. Sőt ugyanitt megjelent a jókora „Nagy UHU Linux könyv" is.
Így most csak az olyan kényes pontokra térnék ki, amelyek a legtöbb kérdést vethetik fel. Az első ilyen a lemezrészre osztás folyamata. Ha „más" rendszer is van a gépen (és jó hazai szokás szerint mindezt egyetlen lemezrész tartalmazza), akkor egy lemezfelosztó program szükséges, amely képes lemezterületet elvonni a Linux számára. Minderre az egyik legjobb választás a Partition Magic program, amellyel az egész könnyedén megoldható; vagy ha FAT fájlrendszerű a kezelendő lemezrész típusa, akkor az UHU első CD-jének DOS könyvtárában megtaláljuk a Fips nevű apró és ingyenes segédprogramot, amely ugyanezt pillanatok alatt képes elvégezni. Ne ijedjünk meg attól, hogy piciny DOS-os segédprogramról van szó, mert könnyen kezelhető és hibátlanul végzi el a feladatát.
Érdemes kitérni arra is, hogy mekkora legyen ez a lemezrész. Nos, két linuxos lemezrészre lesz szükségünk.
Az egyik a csereterület, a másik a Linux, amely magát a rendszert fogja tartalmazni. (Az elvont lemezterület felosztásánál, ha lehet, először a csereterületet készítsük el.)
A csereterület méretére létezik egy nem teljesen szent szabály, inkább ajánlás, ami a következő: 16, 32, 64, 128 MB fizikai memória esetén a csereterület mérete legyen a fizikai memória kétszerese (tehát 128 MB esetén 256 MB). Ha azonban 256 MB vagy afölötti memória áll rendelkezésre, célszerű egy ezzel megegyező méretű virtuális csereterületet létrehozni. Azt is szokták mondani, hogy 512 MB memória esetén már nem is szükséges csereterületet létrehozni. Én viszont amondó vagyok, hogy ekkor is hozzunk létre 256 MB-ot, jól jöhet, ha nagyon megterheljük a gépünket, és lassítani semmiképpen sem fogja, legfeljebb üres marad.
További fontos szempont, hogy a csereterület fájl is lehet, de biztosabb módszer, ha lehetőleg a lemez elején, egy külön lemezrészen helyezkedik el. (A lemez elején kisebb a hozzáférési idő, mint a végén, tehát hatással van a sebességre.) Vagyis a telepítés előtt már a linuxos lemezrészek elkészítésekor érdemes figyelembe venni és végiggondolni a csereterület méretét, és ennek megfelelően készíteni el számára a lemezrészt.
A másik gyakori kérdés az, hogy miként telepítsünk, ha a gépünket nem tudjuk CD-ről indítani? Az első telepítő CD-nek szintén a DOS könyvtárában található a `rawrite` DOS-os segédprogram, amely lemezlenyomatfájlokat (image) hivatott hajlékonylemezre helyezni. Az ehhez tartozó lemezlenyomatfájl az első CD ***IMAGES*** könyvtárában található ***uhuboot.img*** néven. Ezzel a megoldással hajlékonylemezről is indítani tudjuk a gépünket és telepíthetjük az UHU Linuxot.
A következő meglepetés viszont a régebbi gépek tulajdonosainak szokott fejfájást okozni. Ha gépünk nem rendelkezik PS/2 egérrel, csak régebbi sorossal, akkor könnyen megeshet, hogy az UHU nem ismeri fel. (Ez azonban csak az 1.0-sra vonatkozik, a cikk írásakor éppen kiadás előtt álló 1.1-esben már nincs benne ez a hiba). Ebben az esetben kézzel kell beállítanunk az egerünket.
A telepítés során a harmadik említésre méltó lépés, hogy a telepítés végeztével a rendszerbetöltőt valahová el kell helyeznünk. Miután bekapcsoltuk a gépet, a rendszerbetöltő fogja elindítani a Linuxot; a telepítő több helyre is

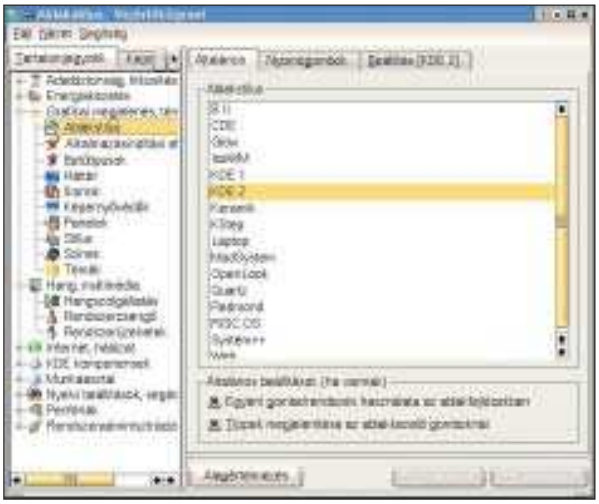
felajánlja a felrakását. Több fizikai lemez esetén még azt is eldönthetjük, hogy melyik lemezen helyezzük el. Ha egyetlen lemezünk van (vagy több lemez esetén az első lemezre telepítettünk), akkor célszerű az „ajánlott" pontot kiválasztani, így a telepítő a rendszerbetöltőt közvetlenül a lemez elejére, a merevlemez rendszerindító területére (Master Boot Record, MBR) rakja. Ha „más" rendszer is ott csücsül a lemezen, akkor is ez a teendő, mert a telepítő intelligensen felismeri, hogy más rendszer is van a gépen, és ennek megfelelően írja meg a rendszerbetöltő bejegyzéseit. Továbbá fontos lehet a rendszerbetöltő milyensége. A szakkönyvek általában a LILO-ról (LInux LOader) írnak, talán csak a legfrissebbek jelentenek ez alól kivételt. Valóban, sokáig a LILO felelt a Linux betöltéséért, ezért a szakkönyvek is ezt ismertették bővebben és az interneten is erről találjuk a legtöbb leírást. Az UHU a korszerűbb és rugalmasabb GRUB rendszertöltőt használja, amelyre a második korong kézikönyve is kitér, valamint a ➲ http://www.uhulinux.hu weboldalon is olvashatunk róla. Ha nem tudjuk, mi a teendő és tanácstalanok vagyunk, célszerű a telepítő által ajánlott megoldást elfogadni.

Az első felfedezőút és az utolsó simítások

A telepítést követően a rendszer azonnal „kulcsrakész", tehát semmiféle teendőnk nincsen. Gyakorlatilag egyszer

sem kell újraindítani a számítógépet, minden átgondolt, gördülékeny. A rendszer betöltődése után mindig be kell jelentkezni: a telepítés során létrehozott felhasználói névvel és jelszóval tudunk felhasználóként belépni. Ezután el tudjuk végezni az utolsó simításokat: a hálózati kapcsolatok kialakítását, a nyomtató hozzáadását és számos egyéb művelet is. Indítsuk el az UHU ***Vezérlőpult*** programot, ami a menüben található. Írjuk be a telepítés során megadott rendszergazdai jelszót (kérni is fogja).
A ***Vezérlőpult*** használata is igen egyértelmű, sőt sorozatunkban sokat fogunk rá hivatkozni. Most inkább azt tekintsük át, hogy az első indítás után mit érdemes elintézni. Az UHU alapértelmezett ablakkezelője a Gnome, bár a KDE sokkal népszerűbb. Igaz, bejelentkezéskor megváltoztathatjuk az ablakkezelőt, de ha a KDE bejelentkezéskezelőjét szeretnénk (`kdm`), akkor a ***Vezérlőpultban*** tudjuk beállítani. A ***Szolgáltatások*** részben keressük ki a `gdm` bejegyzést, és nyomjuk meg a ***ne induljon el*** gombot. Ha megkérdezi, hogy most leállítsa-e a `gdm`-et, akkor a válaszunk nem. Ezután keressük ki a `kdm` bejegyzést, és nála nyomjuk meg az ***elinduljon***

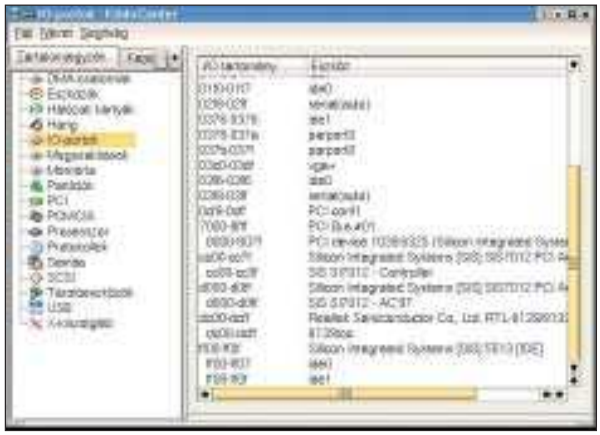
gombot. Meg fogja kérdezni, hogy elindítsa-e a `kdm`-et; a válaszunk szintén nem. Az új beállítás érvényesítéséhez lépjünk ki a bejelentkezőképernyőhöz és indítsuk újra a grafikus felületet. Ezt az ALT+I vagy az ALT+E billentyűparanccsal tehetjük meg, attól függően, hogy feltettük-e a frissítéseket vagy sem.
Amennyiben a hálózati kártyánkat (ha az internetelléréshez szükséges) és a DNS-adatokat (szintén a ***Vezérlőpultban***) beállítottuk, akkor elindíthatjuk az ***UHU tárcsázó*** segédprogramot, amellyel internetkapcsolatunkat felépíthetjük. Jópofa lehetőség, hogyha a tárcsázóval ADSL kapcsolatot építettünk fel, akkor az UHU ***Vezérlőpult/Szolgáltatások*** részében önműködővé tehetjük a feladatot. Itt keressük ki az ***adsl/adsl indítása*** bejegyzést, és kattintsunk az



„elinduljon" gombra. Ezután bármikor, amikor elindítjuk a gépünket, a rendszer betöltődése során az ADSL kapcsolatot is felépíti, így többet nem kell vele foglalkoznunk.

Grafikus felület és adatok

Szükségünk lehet rá, hogy a KDE felületet beállítsuk, esetleg részletesebb adatokat kérjünk a rendszerről.
Az előbbit a KDE ***Vezérlőpultjával*** tudjuk megtenni, ahol beállítható a háttérkép, a betűk, az ablakok stílusa, sőt akár külön témákat is telepíthetünk, ahogyan azt más rendszerek alatt már megszokhattuk.
Ebben a ***Vezérlőpult***-ban a ***Webböngészés*** menüpont alatt állíthatjuk be, hogy a KDE alapértelmezett böngészője, a Konqueror hogyan is viszonyuljon az internetes adatokhoz. Például nagyon hasznos tulajdonsága, hogy a sütikezelést (cookie) finomhangolni vagy tiltani tudjuk.

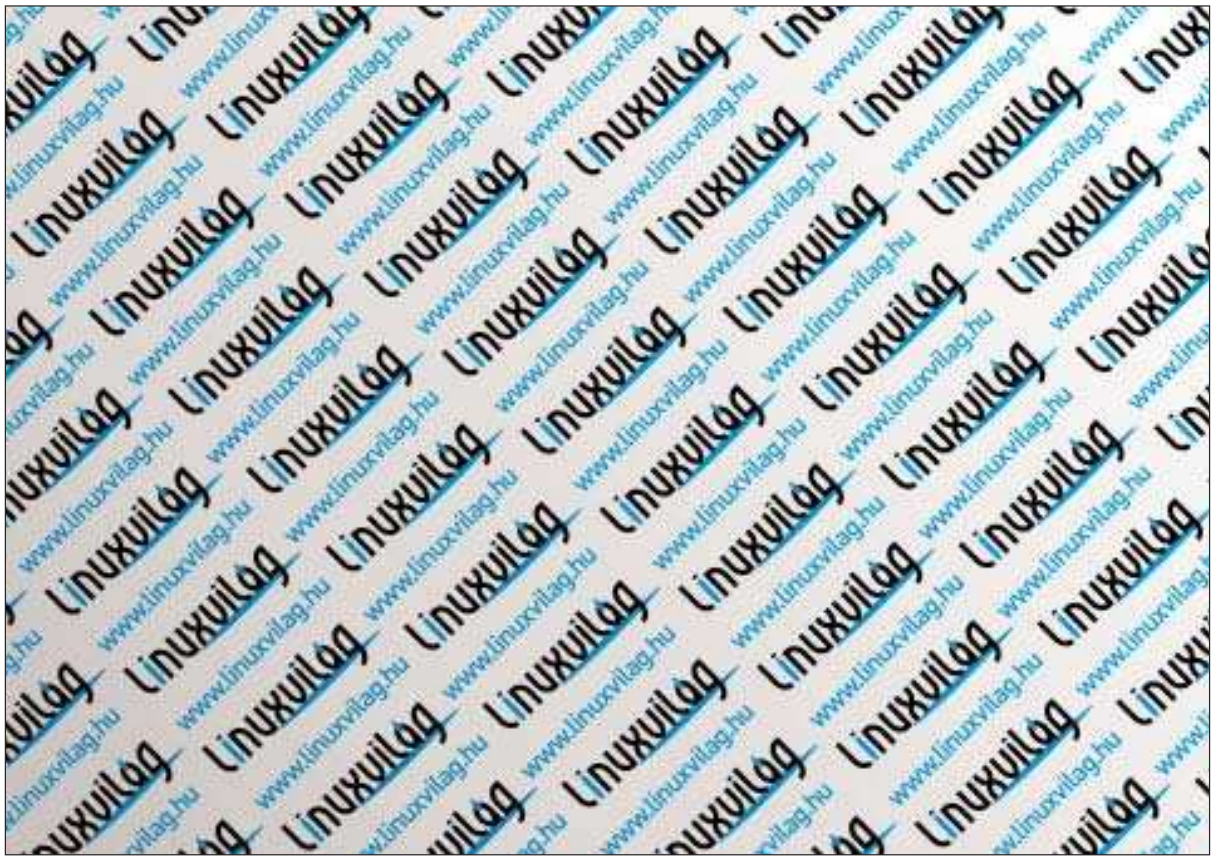


A felugró ablakokat letilthatjuk és mind a Java-, mind a JavaScript-adatokat teljesen kizárhatjuk. Szigorú szabályozással biztonságosabbá és erőforrás-kímélőbbé tudjuk tenni az internetes böngészést. (Biztosan mindenki találkozott már bosszantó felugró ablakocskákkal vagy egy-egy weboldalon túl sok JavaScripttel, ami jelentősen lelassította a rendszerét.)
Itt találhatók a KDE alapvető hangbeállításai, sőt a nyomtatóbeállításokat is módosítani tudjuk a *Perifériák* menüpontban – ennek segítségével hálózati nyomtatót is önműködően keresni tudunk. Alapvető biztonsági és energiakezelési beállításokra is lehetőség nyílik, amelyek rendkívül kényelmes és hasznos eszközök a gép fnomhangolása során. A beállítások elvégzéséhez pedig a KDE *Menü*/*Rendszer* menüpontja alatt található *KInfoCenter* szolgáltathat adatokat, ahol rendkívül részletesen láthatóak az egyes eszközök, a memóriacímek és minden egyéb, amire szükségünk lehet. Hasznosságát nem kell ecsetelnem, ha esetleg egy nem együttműködő (incompatibility) eszközt szeretnénk a rendszerbe beleilleszteni.

Összegzés

Tömören, pusztán a lényegre szorítkozva átvettük, hogy milyen terjesztés az UHU Linux, megvizsgáltuk a gépigényét és a telepítés során felmerülő esetleges buktatókat, valamint rövid körútra indultunk a frissen telepített rendszerben. Megnéztük, hogy mire is használható a *Vezérlőközpont* és hogyan kaphatunk részletes adatokat a rendszerről. Továbbá megemlítettük, hogy hogyan és hol érdemes elvégezni néhány alapvető beállítást. A következő hónapban áttekintünk pár parancsot a konzolos használathoz, telepítünk egy nVidia videokártyát, kitérünk a rendszerfrissítés alapjaira, valamint a KPackage használatának segítségével átvesszük a csomagkezelést. Végül telepítünk egy játékot, hogy olvasóinkat játszani is engedjük.

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



Hálózatok: a TCP/IP hivatkozási modell (3. rész)

Sorozatunk előző részében megismerkedtünk a hivatkozási modellekkel, illetve egyik képviselőjükkel, az OSI modellel. Most a TCP/IP modellel foglalkozunk.

Cikksorozatunk előző részében az OSI hivatkozási modellt mutattuk be, amit a gyakorlatban ugyan sehol sem használnak, mégis nagyon fontos abból a szempontból, hogy szétválasztja a szolgálat, a csatolófelület (interface) és a protokoll fogalmát. Mivel sorozatunk erre a három fogalomra épül, nem árt, ha gyorsan felelevenítjük, mit is jelentenek.

A szolgálat azt határozza meg, hogy az adott rétegnek pontosan mi is a feladata. Ez azt jelenti, hogy a szolgálat nem más, mint olyan elemi műveletek halmaza, amelyet a felső réteg bármikor elvégeztethet vele. Szeretnénk hangsúlyozni, hogy a szolgálat csupán ezeket a műveleteket sorolja fel. Arról azonban, hogy miként hívhatjuk őket meg, illetve hogy az adott feladatot hogyan valósítja meg, mélyen hallgat.

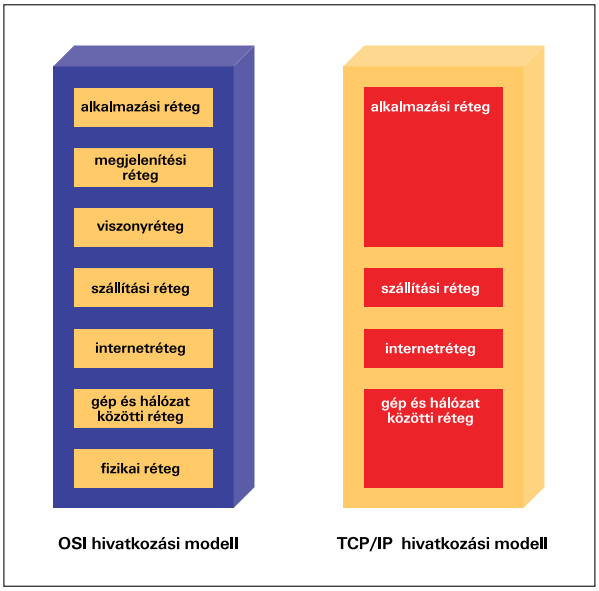
A csatolófelület a szolgálatok elérésének szabályait mondja meg. Ilyen szabályok lehetnek például, hogy milyen értékeket kell átadni, illetve hogy az adott művelet elvégzése után milyen lehetséges eredményeket kaphatunk vissza. A csatolófelület sem árul el semmit a gyakorlati megvalósításról, így ez teljesen rejtve marad a felső rétegek előtt.

A gyakorlati megvalósítás módja a réteg által használt protokolltól függ. Nagyon fontos, hogy az OSI modellben szabadon megválaszthatjuk a használni kívánt protokollt, mivel annak változása nem befolyásolja a csatolófelületet, illetve magát a szolgálatot. (Ez az egész nagyon hasonlít az objektumközpontú programozáshoz: a rétegek egy-egy objektumnak felelnek meg, a szolgálat az adott objektumban található eljárások összessége. A csatolófelület az eljárások értéklistája, illetve a visszatérési értékek típusa; a protokoll pedig az eljárások törzse, amely teljesen láthatatlan a külső szemlélő számára).

A TCP/IP hivatkozási modell

A TCP/IP hivatkozási modell képezi az alapját egy világméretű hálózatnak, amelyet mi már csak internet néven emlegetünk. Ez a modell bizonyos szempontból nagyon hasonlít az OSI-ra, de el is tér tőle. Minderről egy kicsit később szólok, először nézzük meg a modell kialakulásának a körülményeit!

Az OSI modellhez hasonlóan a TCP/IP is rétegekből áll, de csak négy darabból épül fel (lásd az *1. ábrát*) – az alsó két réteg ugyanis „összevontan” van jelen, a viszony- és a megjelenítési réteg pedig nem szerepel a modellben. Legalul a gép és hálózat közötti réteg található. Érdekes módon erről a TCP/IP nem mond semmit, csak azt, hogy itt egy úgynevezett IP-csomagokat továbbítani képes protokollnak kell szerepelnie. Az sincs megkötve, hogy ennek a protokollnak mindenhol ugyanolyannak kell lennie – ez hálózatonként, illetve gépenként is változhat. Mi most ezzel a réteggel nem is foglalkozunk, mivel hivatalosan nem tartozik a TCP/IP-hez, de a következő részben visszatérünk rá, amikor az adatok fizikai átvitelével kapcsolatos kérdéseket tárgyaljuk.



1. ábra Az OSI és a TCP/IP hivatkozási modell összehasonlítása

A modellben szereplő legalsó réteg az internetréteg. Feladata „csupán” annyi lenne, hogy a csomagokat eljuttassa egyik helyről a másikra. Valójában ez egy rendkívül összetett feladat, mivel a célállomás általában nem ugyanannak a hálózatnak a része, mint ahonnan mi küldjük a csomagokat. Lehet, hogy a célpont egy másik földrészen helyezkedik el és rengeteg más hálózaton kell keresztüljuttatnunk a csomagokat.

Fontos, hogy a TCP/IP modell rétegei sem árulják el, hogy miként dolgoznak. A felső rétegeknek csak át kell adniuk a csomagot az internetrétegnek, anélkül, hogy bele kellene gondolniuk, hogy miként jut el a csomag a kívánt helyre.

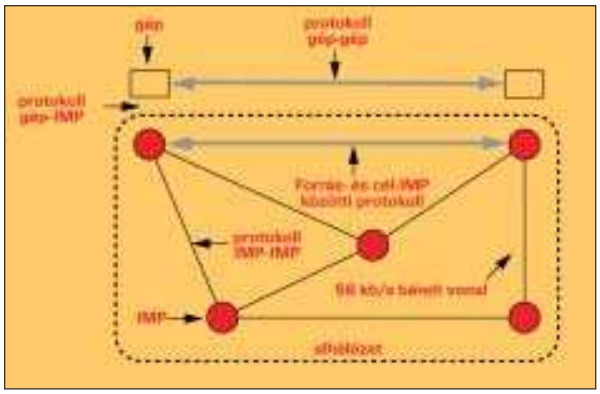
Ez olyasmi, mint a hagyományos postai levelezés. Nekünk csak be kell dobnunk a megcímzett borítékot a postaládába – innentől kezdve a levél sorsa a postára lett bízva. Mi és a levelezőpartnerünk csak annyit veszünk észre, hogy a bedobott levél pár napon belül a címzett postaládájába érkezik. Arra pedig csak következtetni tudunk, hogy mindeközben a levél hány postahivatalon és központon ment keresztül.
Az internetréteg protokollját Internet Protocolnak, röviden IP-nek nevezzük. Mivel itt a csomagok sikeres célba juttatása a legfontosabb feladat, ezért az IP a leghatékonyabb útvonal kiválasztásával, illetve a „forgalmi dugók" elkerülésével foglalkozik.
A TCP/IP hivatkozási modell következő rétege a szállítási réteg, amely gyakorlatilag megegyezik az OSI-féle szállítási réteggel. Ez teszi lehetővé a kapcsolatban résztvevő két fél (pontosabban a társentitások) párbeszédét. Ez a réteg kétféle protokollt használhat: a megbízható TCP-t és a kevésbé megbízható UDP-t.
Ennek a modellnek az IP mellett a TCP (Transmission Control Protocol, azaz átvitelvezérlő protokoll) a második legjelentősebb protokollja, ami egy hibamentes átvitelt biztosító, összeköttetés alapú protokoll. Az átvitel bájtos formában történik, azaz a réteghez beérkező bájtfolyamot előre meghatározott méretű üzenetekre bontja, és ezeket továbbítja az internetrétegnek. A célállomáson ennek a fordítottja zajlik: az internetrétegtől kapott üzeneteket egyetlen bájtfolyammá fűzi össze.
Ahhoz, hogy mindez megvalósuljon, a TCP-nek forgalomirányításra is képesnek kell lennie. Ez annyit tesz, hogy az üzeneteket legfeljebb olyan sebességgel adagolja, ahogy azt a vevő fogni képes. Enélkül a gyorsabb gépek könnyen eláraszthatják üzenetekkel a lassabbakat, teljesen használhatatlanná téve őket.
A réteg másik protokollja az UDP (User datagram Protocol), amely a TCP-vel ellentétben nem összeköttetés alapú, továbbá teljesen megbízhatatlan. Mégis nagy jelentőséggel bír akkor, ha sokkal fontosabb a gyors, mint a pontos válasz. Gondoljunk csak a hálózaton keresztüli hang- és képátvitelre: a felhasználókat kevésbé zavarja, ha egy-két pixel hibásan ér át, mintha a lejátszás során végig akadozna az anyag.
A legutolsó réteg az alkalmazási réteg, ahová a magsabb szintű, a felhasználók által is használt protokollok tartoznak; ilyen például a fájlátvitelre alkalmas FTP, a levelezésre használt SMTP, a virtuális terminál (TELNET) és a névfeloldásért felelős DNS. Ide tartozik még a webböngészők és kiszolgálók között használt protokoll, a HTTP is. Ezek működésével sorozatunk befejező részeiben részletesen is megismerkedünk.

Gondok a TCP/IP-vel

A mai világban a TCP/IP protokolljait használják a leggyakrabban, ami mégsem jelenti azt, hogy ez a modell tökéletes lenne. Kezdjük azzal, hogy a TCP/IP hivatkozási modell eleve nem választja el szembetűnően a szolgálat, a csatolófelület és a protokoll fogalmát; erre a későbbiekben egy példa során rá is fogunk mutatni. A későbbiekben látni fogjuk, hogy nagyon nehéz megállapítani, hogy mi tartozik a szabványhoz és mi a megvalósításhoz. Ezzel szemben az OSI modell nagy gondot fordít arra, hogy ez a két dolog egyértelműen megkülönböztethető legyen.
A modellel kapcsolatos másik gond az, hogy nem általánosítható. Ez azt jelenti, hogy ha ez alapján kéne megterveznünk egy hálózatot, viszont nem a TCP és az IP protokollokat szeretnénk használni, elég nehéz dolgunk lenne. Megfordítva: ez a modell csak a TCP/IP protokollkészlethez alkalmazkodik.
Mindemellett a TCP/IP hiányos modell, mivel nem foglalkozik az adatátvitel fizikai megvalósításával. Nemcsak hogy nem különbözteti meg a fizikai és az adatkapcsolati rétegeket, de meg sem említi őket. Egyszerűen csak egy csatolófelületet ad hozzá, és azt mondja, hogy itt egy olyan valami legyen, ami ezt és ezt a csomagot eljutattja ehhez és ehhez a hálózati ponthoz. Ez azért gond, mert a hálózati és az adatkapcsolati réteg alapvetően két különböző dologért felelős. A hálózati réteg az adatokat az egyik géptől a másik (szomszédos) gépig juttatja el, például vezetéken vagy rádióhullámok, esetleg fényjelek segítségével. A hálózati réteg számára azonban minden adat csak egy bitsorozat. Az adatkapcsolati réteg ezzel szemben már keretekkel működik (lásd az előző részt), azaz gondoskodik arról, hogy az átküldött bitsorozat szétválasztható legyen. Ezenkívül különböző hibajavító kódokkal is dolgozik, amelyek a hibátlan átvitelt biztosítják.
Az IP és a TCP szerencsére jól átgondolt és remekül működő protokoll, a többire ez azonban már nem teljesen igaz. Ezek közül a legtöbbet olyan módon írtak meg, hogy mindenféle átgondolás vagy tervezés nélkül egyszerűen belevágtak. Emiatt azoknak, akik később meg akarták fejteni a működésüket, éjszakákon át a megvalósításban kellett molyolniuk. A megértésre azonban nem is volt mindig szükség, mivel a megvalósítások ingyenesek voltak, s így semmi sem akadályozta meg, hogy ezeket a protokollokat beépítsék az operációs rendszerek legbelső részébe. Ezáltal rendkívül nehézkessé vált a továbbfejlesztésük, illetve a lecserélésük. Ez megmagyarázza azt is, hogy a körülbelül negyed évszázaddal ezelőtt született TELNET protokoll – amely sem az egér, sem a grafikus felület kezelésére nem képes – miért lehet máig is elfogadott (habár ma már inkább „titkosított változatát", az `ssh`-t használják).
Ebből az egészből tulajdonképpen csak egy tanulságot vonhatunk le: míg az OSI modellel rendkívül egyszerűen és tanulságosan írhatjuk le a mai hálózatok működését, addig a TCP/IP erre nem igazán alkalmas. Az OSI modell protokolljainak megvalósítása azonban rendkívül nehéz, így a való életben nem is váltak népszerűvé. A TCP/IP protokollokat széles körben használják, viszont a TCP/IP mint hivatkozási modell valójában semmire sem használható, mivel nem ad teljes elméleti leírást a hálózatokról.

Az internet és a TCP/IP kapcsolata

Az internet eredete körülbelül a 60-as évek közepéig nyúlik vissza. A legismertebb történet szerint az internet egy katonai célokra, pontosabban egy atomtámadás túlélésére „kiképzett" hálózatból fejlődött ki. Ez részben igaz is, és a dologhoz valóban elég sok köze volt az amerikai védelmi minisztériumnak, ám a valóság – mint mindig – ennél sokkal árnyaltabb.
Abban az időben három különálló intézmény is ugyanazon a feladaton, a csomagkapcsolt hálózatok létrehozásán fáradozott anélkül, hogy bármit is tudtak volna egymásról.



2. ábra A kezdeti ARPANET hálózat

Ezek közül csak az egyik, a Rand Corporation által vezetett kutatások voltak katonai indíttatásúak. Ennek az a története, hogy az amerikai védelmi minisztérium úgy gondolta, hogy egy atomtámadás esetén az akkori vonalkapcsolt távbeszélőrendszer összeomlana, így létre kellene hozni egy másik, az ország távoli pontjait összekötő hálózatot, amelyen továbbíthatnák a parancsokat.

A minisztérium ezért kutatási részlegükhöz, az ARPA-hoz fordult. Ez egy meglehetősen érdekes intézmény volt, azért hozták létre, hogy valamennyire kiegyensúlyozzák az akkori Szovjetuniónak az űrkutatás területén megszerzett fölényét. Ugyanakkor az ARPA-hoz nem tartoztak sem tudósok, sem laboratóriumok; úgy működött, hogy ösztöndíjakat és szerződéseket kínált fel egyetemek, illetve cégek számára, és kutatási eredményeiket megpróbálták hasznosítani a haditechnika terén. Így került képbe a Berkeley Egyetem és a Rand Corporation is, ahol számos olyan ember dolgozott, aki meg volt győződve róla, hogy a csomagkapcsolás lehet a feladat megoldása.

A csomagkapcsolás akkoriban eléggé forradalmi ötletnek számított. A kiinduló elgondolás az volt, hogy az elküldeni kívánt üzenetet csomagokra kell bontani. Minden csomag tartalmazta az útvonalválasztáshoz szükséges adatokat is, illetve azokat a dolgokat, amelyek a hibák felismeréséhez és kijavításához szükségesek voltak.

Ezzel a témával a Randon kívül a Massachusetts Institute of Technology (MIT) és a brit National Physical Laboratory is foglalkozott. A három kutatócsoport először 1967-ben találkozott egymással, ekkor hozták létre az ARPANET-re keresztelt hálózat első tervét (*2. ábra*). A gépeket összekötő alhálózat úgynevezett csomóponti gépekből állt. Minden ilyen gép legalább másik kettőhöz csatlakozott, hogyha egy csomópont esetleg ki is esne, a csomagok akkor is célba érjenek – igaz, más útvonalon.

A terv bemutatása után a megvalósítás céljára az ARPA pályázatokat írt ki, amit a BBN nevű cég meg is nyert. Ők telepítették az első csomópontokat, illetve írták meg a hálózati programot. Az ARPANET 1969-ben indult útjára egy NCP (Network Control Protocol) nevű protokollal, amely a TCP elődjének számít.

Az ARPANET hamar népszerű lett; a sikere egyrészt annak tudható be, hogy a kiépítés nyílt volt, azaz független hálózatokból épült fel, amelyek az ARPANET-en belül egyenrangúak voltak egymással. Az idő előrehaladtával egyre több hálózatot kapcsoltak be az ARPANET-be.

Az NCP-nek azonban két súlyos hibája is akadt: nem lehetett a csomópontok mögé címezni és csomagvesztés esetén a protokollok általában lefagytak. Ezért ***Kahn*** és ***Cerf*** javaslatott tett egy új protokoll létrehozására, amelyet TCP-nek neveztek. (Ez még nem a jelenleg használatos TCP, hanem a TCP/IP egyfajta elődje volt). Az új protokoll tervezésekor alapvető szempont lett az, hogy egy új hálózat bekapcsolásakor ne kelljen magát a hálózatot módosítani. Fontos volt az is, hogy a hálózat ne igényeljen globális ellenőrzést, így a hálózatnak nem kellett törődnie azzal, hogy a csomagok célbaérnek-e vagy sem. Ezzel csak a kapcsolatban közreműködő két félnek volt gondja: ha nem érkezett meg a várt csomag, újból el kellett küldeni.

Mivel a TCP-t nem a legalsó szintre valósították meg, fájlátvitelre vagy távoli gépekre való bejelentkezésre könnyen igénybe lehetett venni. Ugyanakkor alkalmatlan volt olyan feladatokra, ahol inkább a kapcsolat sebességére volt a fontos, semmint az, hogy a hálózaton elvesztett összes csomagot pótoljuk. Ilyen feladat volt például a beszédátvitel. Ezért a TCP-t két részre osztották: a TCP-re és az IP-re; a régi TCP-t tehát e két szétválasztott protokoll kettőse, a TCP/IP adja.

A Berkeley egyetem munkatársai eközben egy kényelmes csatolófelületet fejlesztettek ki, amely a foglalat (socket) fogalmára épül (erről részletesen a későbbi részek során lesz szó). Ehhez nagyszámú felhasználói alkalmazást is készítettek, amelyek megkönnyítették az ARPANET használatát. Ez a későbbiekben jelentősen hozzájártult az ARPANET sikereihez, mivel az egyetemek egynél akkoriban kezdtek el több számítógépet is működtetni. Többnyire VAX-okat vettek és egyből ki is építették hozzájuk a helyi hálózatukat. A gond az volt, hogy akkoriban nem volt olyan operációs rendszer, amely a hálózati feladatokat is elláthatta volna. Szerencsére pont ekkor jelent meg a 4.2 BSD, amely elsőként tartalmazta a TCP/IP megvalósítását. Ezek mellett tartalmazta a foglalatokat és több tucat hálózati segédprogramot. Az egyetemeknek ez kapóra jött, és mivel a TCP/IP-vel könnyedén lehetett a helyi hálózatot (LAN) az ARPANET-hez kapcsolni, ezt sokan meg is tették.

1983. január 1-jétől az ARPANET hivatalos protokollja tehát a TCP/IP lett, és egyre több helyi hálózat csatlakozott hozzá. Mivel egyre több gép volt, szükségessé vált a DNS (Domain Naming System) kifejlesztése, amely lehetővé tette, hogy a gépeket ne csak IP-cím, hanem nevek alapján is el lehessen érni.

Eközben a védelmi minisztérium az ARPANET-ről leválasztotta a katonai hálózatokat, és számukra egy különálló, zárt hálózatot hozott létre, ez volt a MILNET. Akkoriban ez a lépés csaknem a gépek felének a leválasztását jelentette. Az ARPANET-et a 80-as évek közepétől már internetként (mint hálózatokat összekötő hálózatot) tartották számon.

A következő résztől elkezdünk a fizikai réteggel foglalkozni, azaz megismerjük az adatátvitel elméletét és gyakorlatát.

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

Hogyan térjünk át Linuxra lépésről lépésre? (4. rész)

Irodai programcsomagok nyújtotta lehetőségek Linux alatt.

A számítógép-használat során a leggyakrabban szövegszerkesztéssel, táblázatkezeléssel és bemutatókészítéssel foglalkozunk. Mi sem bizonyítja ezt jobban, mint az, hogy a legtöbb számítógép-kezelői tanfolyam oktatási anyaga is pont erre a területre irányul. Elmondhatjuk tehát, hogy időnk nagy részét irodai programcsomagok használatával töltjük, így elvárható, hogy rendszerünk kifogástalanul, minden igényt kielégítően szolgálja ki a felmerülő igényeket. Ezt a Linux háza táján is pontosan tudják: a fejlesztők igyekeznek a kedvünkben járni, ennek következtében a kezdeti lemaradást gyorsan sikerült behozni. Ma már nagyon hatékony és szabadon hozzáférhető programok állnak rendelkezésünkre, amelyeknek a használata mindössze alig különbözik már jól megszokott windowsos társaikétól.

Ingyenes, naprakész, hatékony

Ha marketinges lennék, ezzel a három szóval tudnám leginkább jellemezni a linuxos irodai programkínálatot. Gondolom, az ingyenesség előnyét nem kell hosszan ecsetelnem, hiszen mindenki pillanatok alatt el tudja képzelni. Ennek ellenére mégis érzékeltetni szeretném egy kicsit. Most ne feltétlenül otthoni felhasználóként gondolkodjunk, hanem egy kis- vagy középvállalkozás informatikai vezetőjeként. Ilyen környezetben már messze nem mindegy, hogy mire adjuk a fejünket. Esetleg vállaljuk, hogy az összes gépre méregdrága irodai csomagot vásárolunk, s emellett természetesen azt az operációs rendszert is megvesszük, amelyik elengedhetetlen a programcsomag-futtatáshoz. Vagy úgy gondoljuk, hogy egy fillér kiadás nélkül a Linuxot választjuk a terjesztés mellett fellelhető, abszolút versenyképes irodai programokkal, s a megtakarított összeget (ami 10–15 gép esetében már milliós nagyságrendű) a géppark korszerűsítésére tesszük félre. Az utóbbi esetben – természetesen, ha a munka hatékonysága nem romlik – a főnökünk valószínűleg igencsak meg fogja veregetni a vállunkat. Emellett magánemberként sem mindegy, hogy mennyit áldozunk a méregdrága vas megvétele után arra, hogy azt még használni is tudjuk. Ami a naprakészséget illeti: szinte heti rendszerességgel érhetők el az újabbnál újabb frissítések, foltok, valamint a termékek életciklusa is sokkal rövidebb, mint például a Microsoft által fejlesztett Office programcsomag esetében. A hatékonyság tekintetében is sikerült utolérni a versenytársat, de erről részletesebben az egyes programok bemutatása során szólok majd.

Lássuk a medvét!

Avagy nézzük meg, miből is gazdálkodhatunk. A csomagok területén két nagy képviselővel találkozhatunk: az egyik a KDE munkakörnyezethez fejlesztett KOffice csomag, a másik pedig a Sun által elkezdett, de azóta már átalakult OpenOffice.org nevű irodai együttes. Ezenkívül természetesen léteznek még önálló elemek is, mint például a GNUMeric, amely a Gnome grafikus felület táblázatkezelője, vagy a Spreadsheet nevű egyszerű programocska, de nem maradhatnak ki a felsorolásból az olyan Word-klónok sem, mint például az AbiWord.
A mindennapos használat során célszerű azonban a teljes csomagokat használni (lehetőleg egyfélét), mert ekkor megvan az az előnyünk, hogy az egyes összetevők „beszélgetni" tudnak egymással, például táblázatokat helyezhetünk át dokumentumokba, bemutatókba. A két nagy versenytársat összehasonlítva azt látjuk, hogy míg a KOffice beépített, de viszonylag egyszerű megoldást kínál a felhasználóknak, addig az OpenOffice.org inkább a vállalati környezetben is kiválóan használható nagyméretű, széles körben használható, ugyanakkor egy kicsit lassúbb programcsomag. A továbbiakban ezt a két csomagot szeretném bemutatni, ám a kísérteties hasonlóság miatt – amely a windowsos programokra emlékeztet minket – a használatukról nem szólnék részletesen, ezzel ugyanis senkinek sem lehet gondja a későbbiek során.

KOffice

Csakúgy, mint a már megszokott rendszerek, ez az összeállítás is az alapvető, gyakori elemeket tartalmazza: a KWord nevű szövegszerkesztőt, amely szinte az összes jelenleg használatos fájlformátummal (beleértve a Word dokumentumformátumot is) megbirkózik; a KSpread nevű táblázatkezelőt, amely szintén kezeli a versenytársak állományait; a KPresenter nevű bemutatókészítőt (presentation); valamint a Kivio nevű diagramrajzoló programot. Ezenkívül olyan kiegészítők tartoznak ide, mint a KFormula nevű egyenletszerkesztő, a Karbon vektorgrafikus rajzoló, a KChart diagramrajzoló vagy a Korganizer névre hallgató határidőnapló. Ezek mindegyikéről elmondható, hogy külsejét és használatát tekintve a hagyományteremtő Microsoft Office programok másolata, ezáltal nyugodtak lehetünk afelől, hogy a használat során nem ütközünk nehézségekbe. Külön figyelmet érdemel, hogy minden összetevő ismeri a neki megfelelő windowsos fájlformátumot is, tehát a

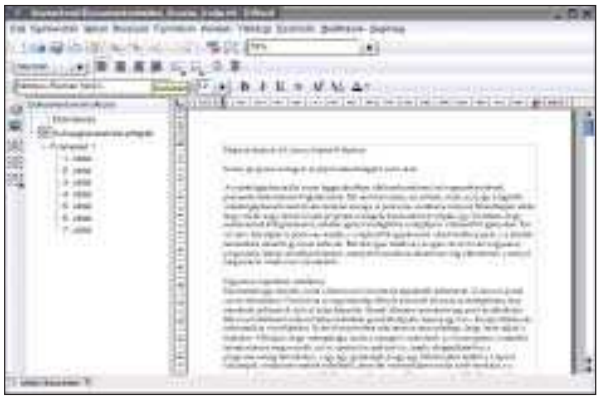
hordozhatósággal sem lesznek gondjaink, bár a bonyolult függvényeket használó táblázatoknál vagy az összetett elemeket tartalmazó dokumentumoknál támadhat némi nehézségünk – ezek azonban az esetek többségében egy-két kattintással orvosolhatók.

Telepítés

A jó öreg SuSE-terjesztésnél maradva álljon itt néhány sor telepítési útmutató gyanánt: nincs más dolgunk, mint rendszerünket a YaST segítségével a programcsomagot alkotó összetevőkkel bővíteni. A YaST csomagkezelő (***Szoftver telepítése, eltávolítása***) moduljával keressünk rá a ***koffice*** kifejezésre, és a találati listában válasszuk a koffice nevű csomagot. Ez önmagában csak az összeállítás közös fájljait és a függvénytárakat tartalmazza. A lista aljában találhatók a valódi összetevők: ***koffice-illustration***, ***koffice-presentation***, ***koffice-spreadsheet***, ***koffice-wordprocessing***. Ha ezeket kiválasztottuk, kattintsunk az ***Elfogad*** gombra, majd hagyjuk jóvá a rendszer által mutatott függőségi listát, s kezdődhet a telepítés. A telepítés befejeztével a KDE menü ***Irodai alkalmazások*** menüpontja alatt fellelhető programok listája egy-két elemmel gazdagodott; próbaképpen bátran indítsuk el őket!

KWord

Elindítás után egy Word kinézetű felület köszönt bennünket. Alapértelmezetten egy ablak pattan fel, amelyen fájlt nyithatunk meg vagy sablonon alapuló új dokumentumot hozhatunk létre. Ha egy kicsit jobban szemügyre vesszük



1. kép A KWord szövegszerkesztő

a ***Dokumentum megnyitása*** fület, azonnal megtapasztalhatjuk, hogy hányféle formátumot kezel. Az sem gond tehát, ha valakitől másfajta szöveget kapunk, nekünk attól még nem kell lemondanunk a program használatáról. A program egyébként a legtöbb megszokott feladatot ellátja: képeket szúr be, diagramokat, egyenleteket kezel, táblázatok szerkesztését teszi lehetővé, sablonok alkalmazását, felsorolást, számozást támogat, van benne helyesírás-ellenőrző, kezeli a fej- és láblécet, körlevelet készíthetünk vele, s mindezt a már jól megszokott formában. Kedvezőtlen vonásaként fel lehet róni, hogy ezeket nem olyan kifinomultan teszi, mint windowsos társa, ám mint tudjuk, ott sem használjuk minden nap a különleges lehetőségeket, így azt mondhatjuk, hogy a mindennapos szükségleteket általában bőségesen kielégíti.

2. kép A KSpread táblázatkezelő

KSpread

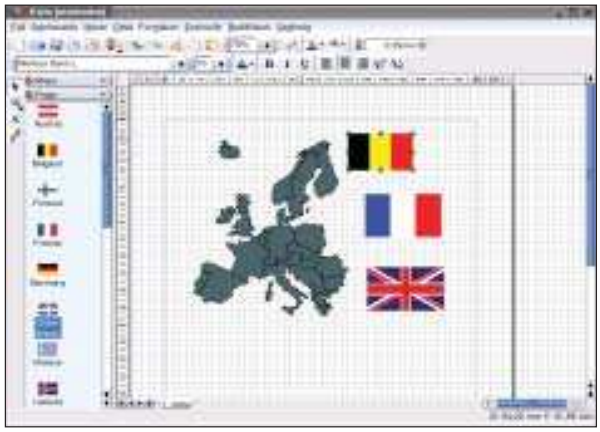
A KOffice táblázatkezelőjéről nagyjából ugyanazok mondhatók el, mint a KWordről. Szinte mindent tud, de csak a maga szerény módján. Hiányzik például az a szolgáltatás, hogy függvény beszúrásakor egérrel kijelölt tartomány esetében látni lehessen a tartomány határait, viszont mentségére legyen mondva: kifogástalanul működik. A függvénytár is meglehetősen gazdag, viszont a függvényelnevezések tekintetében gond lehet, ha más nyelvű, más formátumú fájlt nyitunk meg (például Excel-dokumentumot). A program egyébként cseresz abatos (compatible) a KWorddel és természetesen a többi összetevővel is: szabadon szúrhatunk be KWord-szöveget a KSpreadbe és fordítva. Hatalmas előnye, hogy szembetűnően gyorsan dolgozik, s higgyük el: kiválóan használható.

KPresenter

Ez a program a Microsoft PowerPoint koffice-os megfelelője, segítségével mindenféle bemutató elkészíthető. Talán ennél éreztem úgy, hogy az általam eddig elkészített bemutatók könnyebben megvalósíthatók voltak, mint PowerPointban. A külső itt is egyszerűbb, de minden elérhető, amire feltétlenül szükségünk van. Minden diához leírást, szöveget fűzhetünk, amely segít nekünk, hogy a későbbiekben is megértsük a diákon feltüntetett csekély mennyiségű adatot. Ez a KPresenterben alapértelmezetten fél képernyőnyi területet kap, kiemelve egyúttal a jegyzet használatának fontosságát. Ezenfelül természetesen a sablonoktól a diák közötti átmenet szerkesztéséig mindent beállíthatunk, s a KPresenter esetében is igaz, hogy sebességében Linux alatt a leggyorsabbak közé tartozik.

Kivio

Ez egy folyamatábra- és diagramrajzoló program, amely egy kissé szokatlan az eddigiekhez képest, hiszen ilyet nem mindegyik irodai programcsomag tartalmaz. A használatát valahogy úgy kell elképzelni, hogy elkészítünk benne egy látványos ábrát, például országok zászlajával, Európa-térképpel és minden egyéb nyalánksággal, majd a művet egyrészt kinyomtathatjuk, másrészt egy KPresenter-diára objektumként beszúrhatjuk, változatosabbá téve ezáltal a bemutatót. Továbbá azt az aprócska tényt sem szabad elfelednünk, hogy nem kell hosszú időn át grafikai programokkal pepecselni, hogy a kívánt látványt elérhessük. A folya-

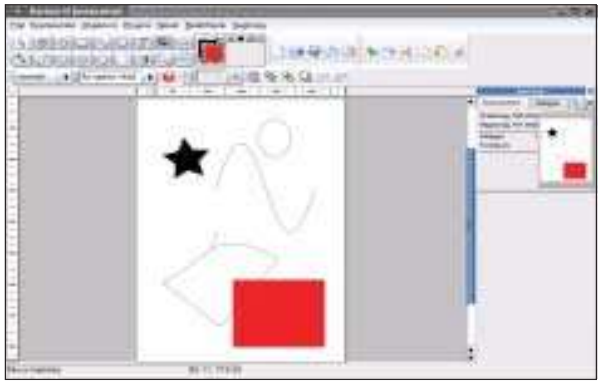


3. kép A Kivio folyamatábra-szerkesztő

matábrák tekintetében azt kell megemlítenünk, hogy kezeli a szabványos jelöléseket: a folyamatábra-alakzatokat, az UML-elemeket, így akár egy programfejlesztő is hasznát veheti, nem beszélve egy házi feladatát leadni készülő diákról. Meg kell hagyni, hogy Windows alatt az ilyesmihez külön program szükséges. A program működése egyébként vektorgrafikus, amellyel nemcsak memóriát takarítunk meg, de a végeredményt is igényesen alakíthatjuk ki. Ami a fájlformátumot illeti, lehetőségünk nyílik az állományokat XML formátumban tárolni, amellyel egy újabb lépést tettünk meg a hordozhatóság rögös útján, így ugyanis olyan szabványos kimenetet határoztunk meg, amelyre más szerkesztőprogramok is támaszkodhatnak. Ezenkívül természetesen saját Kivio fájlformátum is létezik, amely az előző XML leírónyelven alapul.

Karbon14

Ez egy vektorgrafikus rajzolóprogram, amely a jól ismert CorelDraw linuxos megfelelője, s fokozottan igaz rá, hogy nem tud minden szolgáltatást, amit az eredetitől megszoktunk, de kétségkívül jól használható. Ha a Kivióval ötvözzük, gyakorlatilag áthidaljuk a szolgáltatások szegényessége miatt keletkező árkot. A használatáról annyit érdemes tudni, hogy itt is a már bevált normákhoz próbáltak igazodni, semmilyen vad kezelésbeli újítást nem találunk. A kész művet azután a Windows metafájlformátumtól kezdve az Adobe Illustrator által kezelt formátumig bárhogyan menthetjük (beleértve természetesen a már emlegetett XML formátumot is).



4. kép A Karbon14 vektorgrafikus rajzoló

KFormula

Ez a KOffice csomag grafikus egyenletszerkesztője. A programot az esetek többségében nem szükséges elindítani, az egyes összetevők használata során a megfelelő műveletre kattintva (egyenlet, képlet beszúrása) objektumként előugrik; de lehetőségünk van arra is, hogy külön elindítsuk, megszerkesszük vele a képletet, majd a másolás–beillesztés módszerrel átvigyük a céldokumentumba. Ezenkívül a formulát későbbi használat céljából menteni tudjuk – ha például beszámolót és bemutatót készítünk egyszerre, akkor az előre elkészített képleteket beszúrhatjuk a különböző típusú dokumentumokba. A KFormula is támogatja az XML alapú fájlokat, amelyek mellett ugyanazok az érvek szólnak, mint a Kivio esetében.

KChart

A KOffice csomag grafikonszerkesztője. Túlnyomórészt a KSpread programból indul el, de ugyanúgy lehetőségünk van különálló programként futtatni s az eredményt menteni – természetesen itt is választhatjuk az XML kimeneti formátumot. A program használata során egyébként számos grafikontípusból választhatunk és ezek legtöbbjének további altípusai is vannak. Minden aprócska megjelenő szöveget külön-külön beállíthatunk, színeket kombinálhatunk – egyszóval gazdag a beállítási lehetőségek palettája. Az értékek megadása kézzel történik, oszlop–sor (táblázatos) formában. Ez azért is hasznos, mert a KSpread grafikonvarázslóját használva sajnos nem sikerült eredményre jutnom, az ugyanis következetesen „szabálytalan műveletet” hajtott végre.

KOffice – összegzés

Mint láthattuk, a programok nemcsak mintául szolgáló elődeikhez, de egymáshoz is igyekeznek igazodni, mind kinézetükben, mind működésükben. Ezáltal könnyebben sajátíthatjuk el a használatukat, mivel mindet ugyanúgy kell kezelni, mindegyiket egyformán kell megnyitni, az indítás után ugyanazt a sablonkezelő ablakot kapjuk. A működésbeli hasonlóság „mellékhatásaként” pedig kiaknázhatjuk azt a létfontosságú ismeretet, hogy lehetséges az egyes összetevők közti adatcsere. Egymás állományait ugyanis objektumokként kezelik, ezáltal egy nagy-nagy beágyazgatott adathalmazt kezelhetünk, a legváltozatosabb eredményeket érve el. Nagy előnye, hogy az egyes segédprogramok, mint például a diagramszerkesztő vagy az egyenletszerkesztő, akár külön-külön is futtathatók. Van azonban néhány korlátja is a csomagnak: az egyik, hogy az előugró menük hiánya megnehezíti a kezelést, a másik a szolgáltatások terén észrevehető szegényesebb ellátottság. Ez természetesen nem azt jelenti, hogy többre volna szükség; a legtöbb esetben elég, amit a programban találunk, ám ha valami egyedire, különlegesre vágyunk, azt a programcsomag segítségével nagy valószínűséggel nem tudjuk álmainknak megfelelően megvalósítani. További következményként azonban viszont azt tapasztalhatjuk, hogy a program meglehetősen gyorsan fut, s egyszerűségéből adódóan a használata is gyorsabb, mert minden kéznél van, s az egyszerű menüpontok, gombok között gyorsabban is tudunk választani, ami a hosszú, esetleg egyhangúan ismétlődő munka során jelentős időmegtakarítást eredményezhet.

Ha nem elég a KOffice: OpenOffice.org

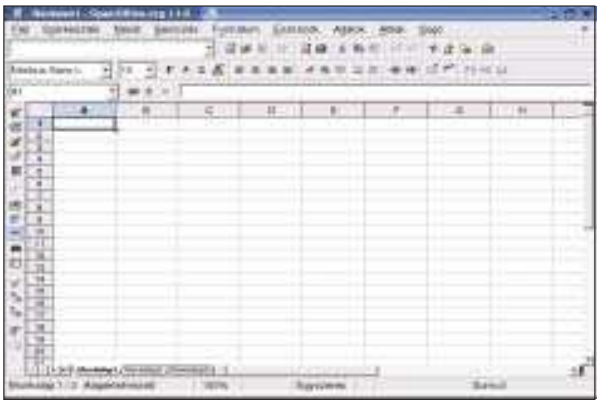
Még mielőtt az időközben kialakult felhasználótábor a kezdetleges megoldásokat hallva elkeseredne (amelyek azért nem olyan kezdetlegesek, inkább úgy fogalmaznék, hogy egy-két változattal le vannak maradva a piacvezető irodai programcsomaghoz képest), gyorsan közhírré szeretném tenni, hogy létezik egy sokoldalúbb, s egyúttal bonyolultabb választási lehetőség is, ami pedig nem más, mint az OpenOffice.org Ez a program a Sun által támogatott StarOffice-ból nőtte ki magát, miután az nyílt forráskódú lett. (Eredetileg teljes irodai munkakörnyezetnek indult ***Start menü***-vel és más egyéb kellékekkel.)
Ez a csomag egyetlen munkakörnyezetnek sem része, teljesen különállóan működik (az X-felületet használva), s a létező változatok segítségével több felületen is használható. Saját grafikus telepítőprogrammal rendelkezik, amelynek köszönhetően mindenféle szakértelem nélkül telepíthető.

A telepítés

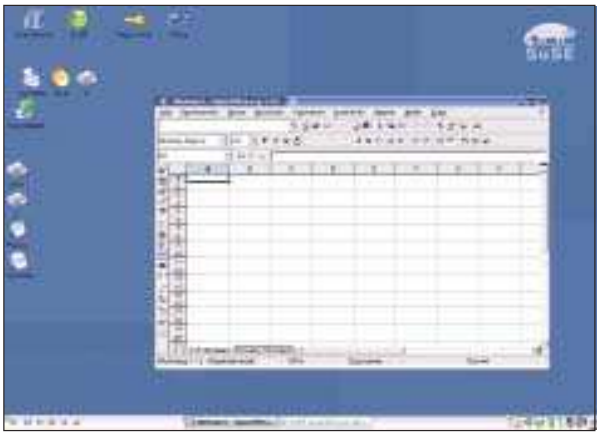
Ha SuSE Linux 9.0-t (vagy 8.2-est) használunk, akkor semmi dolgunk, az OpenOffice ugyanis része az alapértelmezett összeállításnak, így csak használatba kell vennünk. A ***KDE menü/Irodai alkalmazások*** és ***Grafikus programok*** menüpontok alatt találjuk az egyes összetevőket; ám még a használatuk előtt néhány szót szólnék a telepítést illetően azoknak, akik nem SuSE-felhasználók, vagy épp egy régebbi OpenOffice-változat pihen a gépükön, s frissíteni szeretnék. Ehhez nem kell mást tenniük, mint hogy ellátogatnak a ➲ http://download.openoffice.org/index.html címre, és ott kiválasztják a nekik megfelelő OpenOffice.org-változatot, majd azon belül a megfelelő nyelvet, operációs rendszert, s a programot egy tömörített (***tar.gz***) állomány formájában letöltik. Ezt azután parancssorból a `tar -czvf <fájlnév>` paranccsal egy üres könyvtárba csomagolják ki, majd a `./install` parancs kiadásával indítsák el a saját telepítőjét. Ha már volt a rendszerükön egy régebbi OpenOffice-változat, a telepítő ezt önműködően felismeri, s a parancssor képernyőjén visszajelezve régi programunkat az új változatra frissíti. Amennyiben azonban még nem rendelkezünk vele, úgy egy grafikus telepítőprogram indul el, ami érthetően végigvezet bennünket a telepítés folyamatán.

A program

A legújabb változatot elindítva csak egy „ablakkeretet” kapunk, amelyből megnyithatjuk a kiszemelt dokumentumot vagy újat hozhatunk létre. Kezdetnek válasszuk a ***Fájl menü/Új szöveges dokumentum*** menüpontját, s próbáljunk meg a megjelenő ablakban található program és windowsos megfelelője között különbségeket keresni.
A fenti feladat akár fejtörő is lehetne, hiszen az, amit látunk, a megszólalásig hasonlít a Word kinézetére. Működése is csak abban különbözik, hogy sokkal összefogottabb, itt ugyanis mindenféle dokumentumtípushoz (vektoros rajz, táblázat, szöveg) egy programon belül férünk hozzá, s a ***Fájl menü/Új*** menüpontjának segítségével bármilyen típusú dokumentum szerkesztése közben új táblázatot, szöveges dokumentumot vagy épp HTML-dokumentumot nyithatunk meg. Az egyszerűség kedvéért azonban a rendszer menüjébe (***KDE menü***) szétválasztva kerülnek bele, s ha az adott menüpontra kattintva indítjuk őket, az alap-



5. kép A Writer szövegszerkesztő



6. kép A Calc táblázatkezelő

program minden esetben elindul, de a parancs azonnal meg is nyit számunkra a kért dokumentumtípusból egy újat.
Ha már itt tartunk, nézzük meg, hogy milyen elemekből épül fel a programcsomag! Természetesen ez is a klasszikus összetevőket tartalmazza: szövegszerkesztő (Writer), táblázatkezelő (Calc), bemutatókészítő (Impress) és az annyira szokványosnak nem nevezhető rajzolóprogram, a Draw. A rendszer beépített, jól működő helyesírás-ellenőrzőt is tartalmaz, amely önműködő üzemmódban ugyanúgy viselkedik, mint azt már megszokhattuk: a hibásnak vélt szavakat hullámos vonallal aláhúzza. (Itt azonban az önműködő ellenőrzés nem alapértelmezett, az ***Eszközök/Helyesírás-ellenőrzés/Automatikus helyesírás-ellenőrzés*** menüpontban nekünk kell bekapcsolni, függetlenül attól, hogy épp milyen típusú dokumentumot szerkesztünk.) Minden elemről elmondható az, hogy gyakorlatilag az összes megszokott szolgáltatást megvalósítja: gazdagon paraméterezhető, sokszínűen beállítható lehetőségeket kapunk; ám azt az árat fizetjük érte, hogy a program (legalábbis SuSE alatt) elég lassúcska. Nyugodjunk meg, azért még mindig kényelmesen használható.

OpenOffice.org Writer

Az előbbi művelet során is ezt láthattuk működni – tökéletesre sikerült Word-másolattal van dolgunk. Bárhogyan is nézem, semmi újat nem tudnék elsőre elmondani a programról, s hasonlóképpen így vagyok a hátrányok ismerteté-

7. kép A Draw vektorgrafikus rajzoló

sével is. Most már számtalan órája ülök előtte, hiszen e program segítségével készül a jelenlegi cikk (és minden egyéb általam „gyártott" mű is), de nem igazán tudok különbséget tenni a Word és a Writer között. Egy talán mégis akad, amelynek a közelmúltban jó hasznát vettem: a Writer nemcsak a saját formátumait kezeli, hanem a különböző irodai programcsomagok által használatos állományok szinte mindegyikét. Hasonló dolgokról beszéltem fentebb a KOffice esetében is, de ha lehet, itt még bővebb a használható formátumok palettája, s a csomag többi összetevőjéről is pontosan ugyanez mondható el. Akad olyan ismerősöm is, aki az OpenOffice-nak csak ezt az előnyös tulajdonságát használja ki, és átalakítóként (konverter) veszi igénybe.

OpenOffice.org Calc

Ez a program a csomag táblázatkezelője. Ha elindítjuk, azt tapasztaljuk, hogy ez is pontosan úgy néz ki, mint az Excel, s elmondhatom, hogy nagyjából ugyanazt is tudja! Vannak természetesen eltérések és apróbb hiányosságok, de ezt az általános használat során a szerényebb képességű KSpread esetében sem vesszük észre.

OpenOffice.org Impress

A diavetítő és bemutatókészítő program ismét windowsos megfelelőjéhez hasonlít, s a használata is körülbelül annyira bonyolult. Legnagyobb előnye szerintem, hogy jól kezeli a vektorgrafikus objektumokat, így közvetlenül szerkeszthetünk olyan folyamatábrákat, modelleket, amelyeket azután kedvünk szerint alakítgathatunk; nem áll fenn az a veszély, hogy eltorzul az ábra vagy a nyilak épp nem a megfelelő irányba mutatnak. Minden elfogultság nélkül mondhatom, hogy a programcsomag ezen a területen messze lekörözte minden versenytársát, mert ilyen hatékonyan egyikkel sem sikerült előállítanom a kívánt eredményt.

OpenOffice.org Draw

Ez az a program, amely a fenti dicséretet megérdemli. Olyan vektorgrafikus rajzoló, folyamatábra-szerkesztő, grafikonrajzoló program, amely ötvözi a Karbon és a Kivio előnyös tulajdonságait, s a gyakorlatban helyettesítheti a CorelDraw méregdrága programját is, hiszen okos grafikus objektumainak a segítségével szélsebesen dolgozhatunk.

Összegzés

A program ezeken túlmenően beépített képletszerkesztőt tartalmaz, amit szöveges leírónyelvvel is utasíthatunk (nem kell a megfelelő ábrákra kattintani, s ez a programozó hajlamú felhasználóknak általában nagyon tetszik), grafikonok készítését teszi lehetővé a táblázatkezelés során, egyszóval kerek, egész irodai munkakörnyezetet kínál nemcsak az otthoni, de az irodai felhasználók számára is, s ne feledjük, hogy mindezt teljesen ingyen. Ha azonban valaki a bemutatott csomagokat nem szívleli, még mindig szabadon választhat használható szövegszerkesztőket, táblázatkezelőket, bármit, amire szüksége van.
Ismét elmondhatjuk, hogy a Linux ezen a területen is kiállja a próbát, sőt néhány pontban oda is csap az ellenfeleknek, s ezzel gyakorlatilag elérkeztünk oda, hogy új operációs rendszerünket épp olyan jól tudjuk használni, mint annak előtte a Windowst vagy bármelyik felhasználóbarátnak mondott operációs rendszert. A sorozat következő részében az internetes csevegő és üzenő szolgáltatások világában teszünk „áttekintő" kirándulást.

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

Linuxos kiszolgálót mindenkinek! (4. rész)

A SuSE Linux mint kiszolgáló – kisvállalati és otthoni környezetben.

Sorozatunk előző részében beállítottuk a kiszolgáló első tényleges szolgáltatásait az ügyfelek számára, nevezetesen fájlkiszolgálót és DHCP-kiszolgálót létesítettünk, amelyekkel a helyi hálózat támasztotta igények egy részét ki tudjuk elégíteni. Most azt a szolgáltatást nézzük meg, ami a számítógépes hálózatok egyik úttörő szolgáltatása és az internetfelhasználók egyik legkedveltebb, leghatékonyabb információközlő eszköze: az elektronikus levelezést.

Az elektronikus levelezés működéséhez több különböző feladatot ellátó programnak, az úgynevezett ügynököknek (agent) az együttműködése szükséges. Az egyes ügynökök a levelezés folyamán különböző feladatokat látnak el: az MUA (Mail User Agent – Felhasználói levelezőügynök) feladata a levélnek a felhasználói oldalon történő összeállítása, szerkesztése. Ezek gyakorlatilag a felhasználó által használt levelezőprogramok, ilyen a Mozilla Mail, a Pine vagy az Evolution.

Az MDA (Mail Delivery Agent – Levélkézbesítő ügynök) feladata az adott kiszolgálón az őt címzettként megjelölő üzeneteknek a rajta lévő megfelelő postafiókba való eljuttatása.

Az LDA (Local Delivery Agent – Helyi kézbesítőügynök) feladata a helyi címzettnek küldött leveleknek a megfelelő postaládába rendezése. Az LDA a rendszerek többségében megegyezik az MDA-val. Linuxos rendszer alatt ilyen ügynök például a `procmail`.

Utolsóként beszéljünk az MTA-ról (Mail Transfer Agent – Levéltovábbító ügynök), amelynek a leveleknek a megfelelő kiszolgálóhoz való eljuttatása a feladata. Linuxos rendszeren ilyen például a Postfix, a Sendmail vagy az Exim. Jelen cikkünk gerincét a Postfix beállításának az ismertetése alkotja, SuSE 9.0-s rendszer alatt.

A Postfix

A Postfix levélkiszolgáló telepítéséhez indítsuk el a YaST-ot, és a már jól megszokott módon pakoljuk fel a Postfix csomagot. Ha a csomagot telepítettük, a */etc/postfix* könyvtárban lesznek a kiszolgáló beállításait tartalmazó állományok, a */usr/sbin* könyvtárban találjuk meg a kiszolgáló futtatható állományát, valamint a */etc/init.d* könyvtárba kerülnek a kiszolgáló indításához és leállításához használatos parancsállományok.

Továbbá a telepítő létrehozza a */var/spool/postfix* könyvtárat, amelyben a kiszolgáló a feldolgozás alatt álló leveleket fogja tárolni, így a várakozási sorban lévő és a visszautasított levelek is itt lesznek megtalálhatóak. Ez azért fontos, mert ha esetleg rossz beállítással futtatjuk a kiszolgálót, óriási mennyiségű felesleges levéllel töltheti fel a rendszert – e könyvtárak kiürítésével tudjuk a felesleges elemeket törölni, s ezzel megelőzhetjük a rendszer esetleges összeomlását.

Az elektronikus levél útja

Elektronikus levelet minden esetben egy levelezőprogramban (MUA-ban) írunk meg. Amint elküldjük a levelet, egy MTA veszi át kezelésre és továbbításra, méghozzá azon a címen, amelyik a levelezőprogramunkban STMP-kiszolgálóként be van állítva. Ezután a levél fejlécéből az MTA megkeresi a címzettet, illetve a címzettkiszolgálót. Miután azonosította a kiszolgálót, továbbítja neki a levelet. Elképzelhető, hogy a levelünk több kiszolgálón is keresztülhalad (a kiépítéstől függően), mire eléri a címzett gépet. Mivel az elektronikus levelek titkosítás nélkül haladnak keresztül a kiszolgálókon, a köztes kiszolgálókon elvileg lehetőségünk nyílik a levél elolvasására. Ennek kiküszöbölésére születtek meg a különböző titkosítási módszerek, amelyekkel a levél tartalma illetéktelenek számára hozzáférhetetlenné válik. A legkorszerűbb titkosítási eljárások az úgynevezett nyílt kulcsú titkosítási eljárások, amelyek egy kulcspárral üzemelnek, mégpedig olyan módon, hogy a küldő egy mindenki számára ismert kulccsal titkosítja a levelet, amit utána a saját titkos kulcsával csak a címzett tud visszafejteni. Ilyen kulcspárral történő aláírással érhető el az is, hogy a tőlünk származó leveleket elektronikusan aláírjuk, így egyértelműen megállapítható, hogy tőlünk érkeztek-e. Nagyon kell azonban ügyelnünk a titkos kulcsra, mert nyilvánosságra kerülése esetén mind az azonosítási rendszer, mind a titkosítás átjátszható lesz. Miután a levél megérkezett a címzett kiszolgálóhoz, a kiszolgáló LDA ügynöke átveszi a levelet – az ő feladata a levélnek a megfelelő helyi postaládában való elhelyezése. Amint ez megtörtént, levelünket, s a címzett a saját levelezőprogramjával – a MUA-val – el is tudja olvasni ezzel az elektronikus üzenet célba is ért.

A YaST a barátunk

Mint azt már megszokhattuk, a SuSE ismét kínál egy eszközt, amellyel kiszolgálónk alapbeállításait el tudjuk végezni. A YaST ***Hálózati szolgáltatások*** menüpontja alatt található ***Levéltovábbító szerver*** modult indítva lehetővé teszi számunkra az elsődleges levélkiszolgáló szabványos felületen történő beállítását.

A megjelenő ***Kapcsolat típus*** ablakban ki kell választanunk a hálózati kapcsolatunkat. Ha interneten keresztül is szeretnénk levelezni, tehát nem csak egyfajta vállalati levelezést szeretnénk megvalósítani, internetkapcsolatunk típusától függően válasszuk az ***Állandó*** vagy a ***Betárcsázós kapcsolat***-ot. Amennyiben „betárcsázással" kapcsolódunk az internetre, a levelező a várakozási sorba érkező leveleket nem továbbítja önműködően, elindításukat a `sendmail -q` parancs révén kézzel tehetjük meg. Ilyen esetben természetesen írhatunk egy olyan parancsállományt, amely az internetes kapcsolat létrejöttekor lefuttatja a várakozási sor ürítésére szolgáló parancsot.

Amennyiben víruskereső szolgáltatást is szeretnénk kapcsolni a levélkiszolgálóhoz, azt a ***Víruskeresés engedélyezése*** bekapcsolásával tehetjük lehetővé. Ha az Amavis víruskereső nincs telepítve, a YaST ezt is elvégzi helyettünk.

A következő lépésben megadhatjuk a levelek továbbítására használni kívánt SMTP-kiszolgálót. A mező értékének adjuk a `localhost` értéket, amennyiben a jelenleg telepített kiszolgálót szeretnénk használni. Ha azonban egy másik kiszolgálóval szeretnénk a leveleket továbbítani, a teljes internetcímét adjuk meg.

Az ***Álcázás*** gombra kattintva lehetőségünk nyílik gépünk és felhasználóink alapértelmezett nevének és címének a felülírására. Ez akkor lehet hasznos, ha a rendszert internetes és belső hálózati (intranetes) környezetben egyaránt használjuk, valamint több címtartományt is kiszolgálunk, ezért különböző felhasználóinkhoz különböző címek tartoznak.

Az első sorban a ***Doménnév a 'From' (küldő) fejlécmező számára*** mezőben megadhatjuk a kiszolgálóhoz tartozó elsődleges címet, így amennyiben például a gépünk a ***kiszolgalo.tartomany.hu*** címmel azonosítható és mi ebbe a mezőbe a ***tartomany.hu*** értéket írjuk, a ***felhasznalo@kiszolgalo.tartomany.hu*** helyett a kimenő levelek feladója minden esetben a ***felhasznalo@tartomany.hu*** lesz.

A ***Doménnevek helyi kézbesítéshez*** mezőben megadhatjuk, hogy a helyi levelek kézbesítésekor a felhasználó neve után milyen cím szerepeljen. Továbbá lehetőségünk van annak a beállítására is, hogy csak a helyi nevet akarjuk-e álcázni vagy egyéb címeket is felül akarunk-e írni. Ezenkívül azt is megtehetjük, hogy minden egyes felhasználónkhoz egyesével rendelünk hozzá elektronikus levélcímeket. Ehhez a ***Hozzáadás*** gombot szükséges használnunk, és a felhasználót ki kell választanunk a listából, valamint meg kell adni a hozzá tartozó elektronikus levélcímet. Ezzel az ***Álcázás*** menüpont beállítási lehetőségeinek a végére értünk – az ***OK***-ra kattintva visszatérünk a ***Kimenő levelek*** ablakba.

Az ***Azonosítás*** gombra kattintva lehetőségünk van elérési jellemzők megadására a szolgáltatónk által rendelkezésünkre bocsátott SMTP-kiszolgálóhoz, ha a szolgáltató ezt a levelek elküldéséhez kérné. Amennyiben a saját gépünkön (localhost) lévő kiszolgálót használjuk, és előzőleg nem állítottunk be elérési korlátokat, ezeket a mezőket nyugodtan kihagyhatjuk.

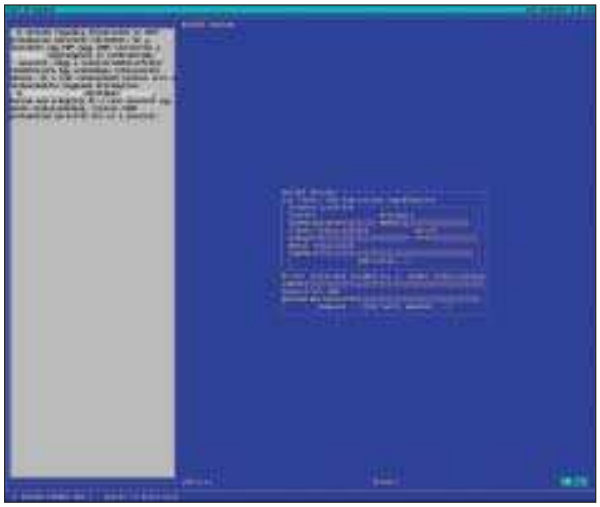
A következő megjelenő ablak a ***Bejövő levelek*** ablaka; itt lehetőségünk nyílik beállítani, hogy a bejövő levelek melyik címzetthez kerüljenek, továbbá azt is, hogy a helyi kiszolgálónk egy másik levélkiszolgálóról töltse le a nekünk szóló leveleket, éppen úgy, mint azt a levelezőprogramok is teszik. Az utóbbinak akkor van értelme, ha kapcsolt vonali internetszolgáltatással rendelkezünk, ezért nincs rá lehetőség, hogy közvetlenül a kiszolgálónak címezzék azokat.

Nézzük először a levelek kézbesítésének a beállításait! ***A root leveleinek továbbítása az alábbi felhasználónak*** változó értékeként adjuk meg azt a felhasználót, akinek a rendszergazdához érkező leveleket továbbítani szeretnénk. Ez azért szükséges, mert a `root`-ot általában nem használjuk levelezési célokra, a rendszer felügyeletét végző embernek pedig valószínűleg egy külön felhasználót hoztunk létre, így neki továbbíthatjuk az olyan leveleket, amiket a rendszergazdának kell megkapnia, például a naplóállományokat, a riasztásokat.

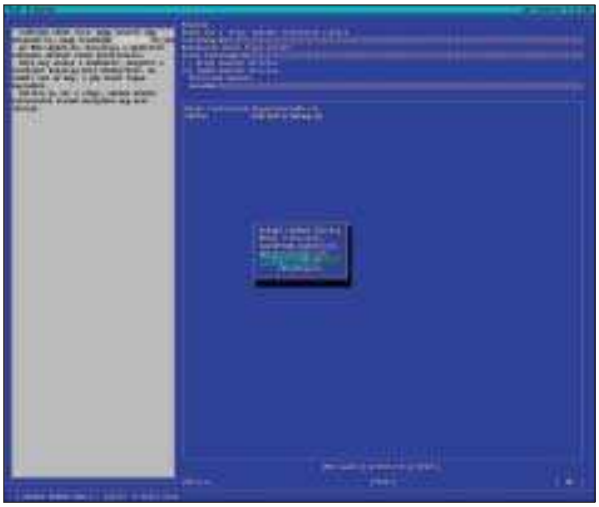
Kézbesítési mód-ként válasszuk a ***procmailen keresztül*** lehetőséget, ami azért lesz hasznos, mert a `procmail` LDA-hoz tartozik egy központi és egy felhasználótól függő beállításállomány, amiben a levelek szűrését, csoportosítását központilag vagy külön-külön is szabályozni tudjuk. Bővebb tájékoztatással a `procmail` súgóállománya szolgál.

Az ***Álnevek*** gombra kattintva szerkeszthetjük a ***/etc/aliases*** állományt, amely a helyi felhasználó–kézbesítési hely párokat tartalmazza. Itt lehet beállítani, hogy az adott felhasználónak érkező levelet a levelező hová továbbítsa. Ez lehet egy másik helyi felhasználó, de egy másik gépen lévő felhasználó is, emiatt itt akár egy elektronikus levélcímet is megadhatunk.

A ***Virtuális domének*** gombra kattintva a ***/etc/postfix/virtual*** állomány tartalmát szerkeszthetjük. Ennek hasonló a szerepe, mint a ***/etc/aliases*** állománynak, de itt a bal olda-

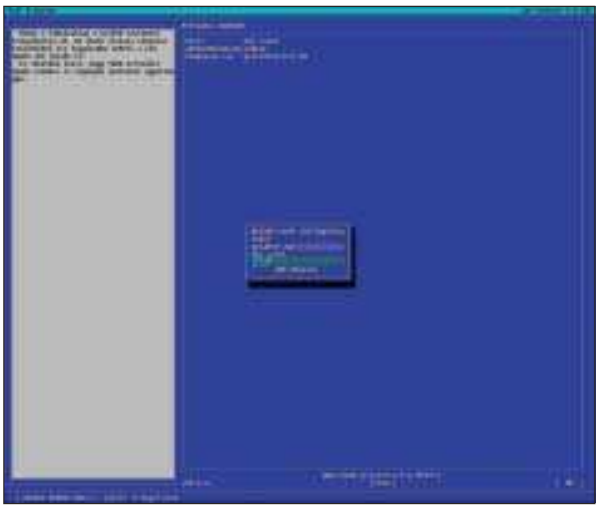


1. kép A bejövő levelek beállítása a YaST-ban



2. kép Álcázás és alapértelmezett címek beállítása a YaST-ban

lon megadhatunk felhasználót, tartományt vagy akár elektronikus levélcímet is. Például beállítható, hogy a ***tartomany.hu*** tartomány bármelyik felhasználójához érkező levelet egyvalaki kapja meg. Lássunk egy-két példát!
Ha egy olyan bejegyzést készítek, aminek a bal oldalán az ***info@tartomany.hu*** áll, míg a jobb oldalon a titkárnő, akkor a fenti elektronikus levélcímre érkező leveleket a titkárnő felhasználó fogja megkapni. Amennyiben olyan szabályt készítek, hogy a ***@tartomány2.hu: főnök***, akkor a ***tartomány2.hu*** tartományba érkező összes levelet a főnök felhasználó kapja meg. További útmutatást a `man virtual` paranccsal kaphatunk. Mind az ***aliases***, mind pedig a ***virtual*** állományokat a fent leírt helyeken kézzel is szerkeszthetjük, de a szerkesztés után az adatbázist is frissíteni kell. Első esetben ezt a `newaliases`, az utóbbiban a `postmap /etc/postfix/virtual` paranccsal tehetjük meg.
Utolsó beállításként nézzünk egy hasznos eszközt azok számára, akiknek különböző szolgáltatóknál vannak postaládái, ideiglenes kapcsolattal rendelkeznek, és így nincs lehetőségük közvetlen levélfogadásra a nap minden percében. A Linux ennek feloldására is kínál egy eszközt, a Fetchmailt. Ezzel a programmal a megfelelő beállításállományban megadott értékekkel csatlakozik a gépünk a távoli kiszolgálóhoz és onnan a megfelelő levelezőprotokoll használatával letölti leveleinket a saját kiszolgálónkra, így olyan módon tudjuk kezelni őket, mintha közvetlenül a kiszolgálónkra érkeztek volna.
Egy ilyen beállítást ki is tudunk tölteni a ***Bejövő levelek*** oldalon, de ha a ***Részletek*** gombra kattintunk, a felbukkanó új ablakban egész listát vehetünk fel a különböző felhasználókhoz. A ***Hozzáadás*** gomb megnyomására felbukkanó ablakban meg kell adnunk a kiszolgáló nevét: ez a távoli postaládánkat tároló kiszolgáló címe lesz; a használni kívánt protokollt, ami általában vagy a POP3 vagy az IMAP, továbbá meg kell adnunk a távoli felhasználó nevét jelszavát, amivel hozzáférésünk lesz a rendszerhez; és végül be kell írnunk a felhasználót, akinek a helyi gépen a leveleket le akarjuk tölteni.
Ha végeztünk a lista elkészítésével, az itt felvett jellemzők a ***/etc/fetchmail.rc*** állományban mentésre kerülnek, amelyben



3. kép Virtuális címek és kézbesítési helyük

a későbbiek során a listát kézzel is szerkeszteni tudjuk.
Az állomány szerkesztésével lehetőségünk van SSL-kapcsolattal is ellátni a letöltést, így a jelszavak és az adatok titkosítva haladnak a távoli kiszolgáló és a miénk között – amennyiben ezt a távoli kiszolgáló számunkra lehetővé teszi. Ehhez a létrehozott bejegyzés végére be kell szúrni az `options ssl` tagot.
Ha ezzel is végeztünk, zárjuk be a varázslót, amely elkészíti nekünk a levélkiszolgáló alapbeállításait.

Nézzük meg a beállításokat közelebbről is!

Bár a varázsló ügyes megoldás a SuSE részéről, korántsem lehet minden apró lehetőséget kiaknázni vele, ezért lehet, hogy kézzel is bele kell nyúlnunk a beállításokba.
Először nézzük a ***/etc/postfix/main.cf*** állományt. Megvizsgálunk pár beállítást, amit a kiszolgáló biztonsága érdekében mindenképpen el kell végeznünk. Az első ilyen a `mynetworks` – ennek a változónak az értéke azt mutatja meg, hogy az SMTP-kiszolgáló mely hálózatoknak nyújt levélküldési szolgáltatást. Ahhoz, hogy elkerülhessük a

reklámozók rajtunk keresztüli történő levélszemétküldését (spam), ezt a változót olyan módon kell beállítanunk, hogy csak a saját gépről (localhost), esetleg a belső hálózat IP-tartományából fogadjon el levelet küldésre. Például a `mynetworks=127.0.0.0/8 10.0.0.0/24` beállítás csak a saját gépről, valamint a 10.0.0.1 és 10.0.0.254 közötti gépekről fogad el levélküldési parancsot. Egyes ügyfeleket adhatunk meg IP-címmel, vagy hálózatokat címmel és hálózati maszkkal, egyszerű felsorolásként, szóközzel elválasztva.
A `myorigin` változó tárolja a kiszolgáló elsődleges tartománynevét. A `mydestination` változóban tartományneveket sorolhatunk fel vesszővel elválasztva, ezeket fogja a kiszolgáló a beérkező leveleknél címzettként elfogadni.
Amennyiben azt szeretnénk, hogy a levél beérkezése után a Postfix a feldolgozást adja át a `procmail`-nek, valamint a felhasználók szűrési szabályokat tudjanak írni, a `mailbox_command = procmail -a "$EXTENSION"` sornak mindenképpen szerepelnie kell a beállítások között.
Amennyiben a rendszer terhelésének ellenőrzéséhez korlátozni szeretnénk a postaládák méretét, illetve meg kívánjuk adni az elküldhető levél legnagyobb méretét, ezt a `mailbox_size_limit` és a `message_size_limit` változók beállításával tehetjük meg. Mindkét változó egy számot vár értékként, ez a szám pedig a méret leírása bájtokban. Például az 1 MB-os postaládaméret beállításához a `mailbox_size_limit=1024000` értéket szokták megadni.
A */etc/postfix/virtual* állomány tárolja a virtuális tartományokhoz tartozó elektronikus levélcímeket és felhasználókat, ha ilyet a varázslón keresztül készített beállításban megadtunk. Amennyiben eddig nem tettük volna meg, a fent leírt szabályokat itt is meg tudjuk adni. Bal oldalon azt a címzettet vagy címtartományt szükséges beírni, ahová a levél érkezik, majd egy szóköz után azt a helyi felhasználót vagy elektronikus levélcímet kell megadnunk, ahová a szabálynak megfelelő levelek továbbítódnak. Így tudunk olyan elektronikus levélcímeket is létrehozni, amelyekhez nem tartozik valóságos felhasználó a rendszerben, hanem csak egy távoli postafiókba továbbítjuk a levelet.
Ne felejtsük el azonban, hogy az állomány módosítása után a `postmap /etc/postfix/virtual` parancsot le kell futtatnunk – ha a *virtual* állomány a */etc/postfix* könyvtárban helyezkedik el – ahhoz, hogy a változásokat a rendszer érvényre juttassa. A mindenkori adatbázis a */etc/postfix/virtual.db* állományban található.
A */etc/postfix/access* állományban állíthatjuk be a szűrési feltételeket az elektronikus levelek feladójára, a teljes elektronikus levélcímre vagy az elektronikus levél feladójának tartományára. Ezzel egy egyszerű levélszemétszűrést állíthatunk be, ami mostanában már sajnos nem túl hatékony, mert a levélszemétküldők vagy állandóan változó címet használnak, vagy úgy választanak címet, hogy a feladót a saját tartományunkból érkezőként állítják be.
A szűrési feltétel egy párosból áll, amely egy mintából és egy cselekvésből tevődik össze.
Minta lehet egy elektronikus levélcím, például a *felhasznalo@kiszolgalo.tartomany.hu* – ekkor a teljes címet figyeljük, vagy a cím töredéke (*felhasznalo@*), amikor is a felhasznalo nevű felhasználóktól érkező levelek kerülnek szűrésre, függetlenül attól, hogy a levél milyen tartományból érkezett. Végül, de nem utolsósorban a szűrési feltétel lehet egy tartomány, tehát például a *kiszolgalo.tartomany.hu* vagy a *tartomany.hu*. Az utóbbi esetben természetesen az első példa is szűrt lesz, de emellett az összes olyan cím is, ami *tartomany.hu*-ra végződik.
Cselekvés lehet a levél eldobása (`DISCARD`), a levél visszautasítása (`REJECT`) vagy a levél elfogadása is (`OK`). Ezeken kívül beállítható még, hogy a beérkező levelet tartalomszűrésnek is alávessük, ehhez a `FILTER` kulcsszót kell megadni. Erről és többi beállításról bővebb leírást a `man access` parancs kiadásával kaphatunk.
A *virtual* tábla frissítéséhez hasonlóan az *access* táblát is frissíteni kell a `postmap` paranccsal, pontosabban a `postmap /etc/postfix/access` paranccsal. Ahhoz, hogy ezt a szűrést használni tudjuk, a */etc/postfix/main.cf* beállításai között szerepelnie kell a `smtpd_sender_restrictions=hash:/etc/postfix/access` beállításnak.
Ezzel a legfontosabb beállításokat talán el is végeztük levelezőrendszerünkön, ami mostanra már működőképes. Ne felejtsük el a tűzfalunkon a megfelelő kapukat kinyitni, ha szükségesek, valamint figyeljünk oda arra is, hogy a YaST-on belül a *Futási szint szerkesztő*-ben szintén be kell állítani, hogy a megfelelő futási szinteken a Postfix önműködően elinduljon. Továbbá ne feledjük, hogyha a beállításokat elvégeztük, a `/etc/init.d/postfix restart` paranccsal indítsuk újra a levélkiszolgálót, hogy az új beállítások érvénybe lépjenek.
Ha végeztünk a kiszolgáló beállításával és elindítottuk, akkor a biztonság kedvéért kövessük figyelemmel a */var/log/mail*, *mail.err*, *mail.warn*, *mail.info* naplóállományokat, hogy a kiszolgálón felmerülő esetleges hibákat mielőbb orvosolni tudjuk. Nincs ugyanis kellemetlenebb dolog, mint egy rosszul beállított kiszolgáló miatt elkallódó levél.
Írásom végére érve elmondhatjuk, hogy működő SMTP-kiszolgálót hoztunk létre. Sorozatom következő részében ezt kiegészítjük azzal, hogy a legnépszerűbb ügyfél-hozzáférési protokollokat telepítjük a rendszerre, foglalkozunk a POP3 és IMAP protokollokkal, illetve azok titkosított, úgynevezett SSL-változatával.

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

Helyreigazítás

A 2003. decemberi számban megjelent cikkemben a rendszer hálózati csatolóinak ellenőrzésére az `IPCONFIG` parancsot neveztem meg – ez elírás, ugyanis a hálózati csatolókat linuxos rendszereken az `IFCONFIG` paranccsal tudjuk listázni. Az `IPCONFIG` is működő parancs, de az egy másik kereskedelmi operációs rendszer hasonló célokat szolgáló utasítása.

A fenti elírás miatt minden kedves olvasó elnézését kérem.

Néhány érv az Xdialog mellett és ellen

Grafikus felhasználói felületek könnyedén és gyorsan héjprogramozással.

Az Xdialog fejlesztése azzal a céllal indult, hogy olyan programot hozzanak létre, amellyel a `dialog` vagy a `cdialog` adta lehetőségeket használó parancsfájlokat módosítás nélkül grafikussá varázsolhatjuk. Az Xdialog az említett alkalmazások összes elemét kezeli és ezek mellett számos újat is bevezet. Az utóbbiak között említhetjük a fájl- és könyvtárválasztót, a fanézetet, a súgógombot – és még sorolhatnám. További előny, hogy az Xdialog a GTK+ eszközkészletet használja, így a segítségével létrehozott grafikus parancsfájlunk alkalmazkodhat a felhasználó által választott asztaltémához. A projekt honlapja a ➲ http://xdialog.dyns.net/ címen érhető el.

Beszerzés és telepítés

Az 57. CD Magazin/Xdialog könyvtárban található a legfrissebb változat, amely a cikk írásának pillanatában a 2.1.1-es. A forrás ***.tar.gz*** és ***.tar.bz2*** formában tömörítve, továbbá SRPM formátumban érhető el. Több felépítésre található előfordított RPM csomag is. A ***.deb*** csomagok egy kicsit lemaradtak az RPM-hez képest, ugyanis a 2.0.6-os volt közülük a legújabb. Az én Debian 3.0 r1 változatomban megtalálható 2.0.5-ös óta a napló tanúsága szerint nem történt sok változás, ezért a kipróbáláshoz is ezt használtam. A forrásból történő telepítés sem egy ördöngösség, a megszokott `./configure && make && make install` biztos eredményre vezet.

Általános használata

```
Xdialog [<GTK+ opciók>] [<általános opciók>]
➥[<átmeneti opciók>] <dobozopció> ...
```

vagy

```
Xdialog <különleges opció>
```

Az első esetben az `[<általános>] [<átmeneti>] <doboz>` sorozat tetszőleges számban ismétlődhet egyetlen Xdialog-sorban – ezt hívják dobozláncolásnak. Az általános és átmeneti beállítások (option) elhagyhatók, de utolsóként minden Xdialog-sorban egy dobozbeállításnak kell szerepelnie.

- Egy `<általános beállítás>` minden utána következő `<dobozbeállítás>`-ra érvényes, amíg egy másik `<általános beállítás>` ki nem oltja a hatását. Ezek a beállítások az ablak címére, a gombok stílusára és hasonlókra vonatkoznak.
- Az `<átmeneti beállítások>` csak a közvetlenül utánuk következő `<dobozbeállítás>`-ra érvényesek. Ezek a beállítások többnyire az adott doboz különleges tulajdonságaira vonatkoznak.
- A `<doboz beállítás>` határozza meg, hogy az Xdialognak milyen elemet kell megjelenítenie. Legalább három értéket vár: egy szöveget, valamint a rajzolandó ablak magasságát és szélességét képpontban megadva.

Amikor a felhasználó egy ablakot bezár, az Xdialog azt az eredményt egy vagy több karakterlánc formájában a szabványos hibacsatornára küldi. Egy általános beállítással ez a szabványos kimenetre is átirányítható.
Az Xdialog visszatérési értéke hordozza magában az információt, hogy a felhasználó hogyan zárta be az utolsó ablakot. Ez lehet 0, ekkor az ***Ok*** vagy ***Igen*** gombra kattintott. 1 esetén a ***Mégsem*** vagy ***Nem*** gombot választotta.

1. lista Mi a neved?

```
#!/bin/bash

DIALOG="/usr/bin/Xdialog"
CIM="Mi a neved?"

NEVED=`$DIALOG --title "$CIM" --stdout
➥--inputbox "Hogy hivnak?" 0 0`

case $? in
    0)
       $DIALOG --title "$CIM" --msgbox
       ➥"Szia $NEVED!" 0 0
       ;;
    1)
       $DIALOG --title "$CIM" --msgbox
       ➥"Talán legközelebb..." 0 0
       ;;
    255)
       echo "Kilépés."
       ;;
esac
```

255-ös visszatérési értéket kapunk, ha valami hiba történt vagy ha az alkalmazást az ablakkezelő rendszeren keresztül zárták be. További értékek is lehetségesek: a `--help` és `--wizard` átmeneti kapcsolók esetén. Dobozok láncolásakor az első olyan doboznál, amelynél a visszatérési érték nem nulla, a lánc megszakad.
Amikor az Xdialogot különleges beállítással használjuk, semmilyen értéket nem vár el, mindössze adattal szolgál a szabványos hibacsatornán. Próbáld ki, mi történik, ha az Xdialognak nem adsz át más értéket, csak a `--print-version`-t!

Szia, világ!

Minden program itt indul. Lássuk, mire képes a `--msgbox` dobozbeállítás!

```
$ Xdialog --msgbox
'Szia világ!' 0 0
```

Ne feledd, hogy az Xdialog program nevének az első betűje nagybetű. A `--msgbox` három értéket vár: az első az a szöveg, amely megjelenik a dobozban. A továbbiak az ablak magasságát és szélességét jelölik ki, ezek értékeként adhatsz 0-t, ekkor az ablak önműködően akkora lesz, hogy a szöveg éppen beleférjen. Ha a szöveg sortörést tartalmaz, az az ablakban is ugyanúgy fog megjelenni. Ezenkívül a `'\n'` különleges karaktert használhatod a sortörések kijelölésére (*1. kép*, lásd a 71. oldalon).

Mi a neved?

Ezek után lássunk egy valódi alkalmazást! Egy olyan parancsfájlt fogunk írni, amelyik megkérdezi a felhasználó nevét, majd kiírja a képernyőre (lásd az *1. listán* és a *2. képen*). Lássuk sorról sorra a parancsfájl működését! Az első sor mindössze a futtatandó parancsértelmezőt határozza meg – ez régi barátunk, a Bourne Again Shell. Az Xdialog nevét érdemes egy változóban tárolni. Előfordulhat, hogy úgy döntünk, nincs szükség a grafikus csillogásra, és megelégszünk a szöveges üzemmód nyújtotta menüvezérelt felületekkel. Ekkor vissza kell térnünk a `dialog` vagy a `cdialog` használatához. Szerencsére ezek többékevésbé megfeleltethetőek egymásnak, a futtatandó program nevét azonban így is át kell írnunk. Ezt könnyebb egyetlen helyen megtennünk, mint az egész programban, ezért fontos az előrelátás.
Mivel több ablakot használunk a programban és szeretnénk ragaszkodni az egységes kinézethez, az ablakok címét is egyetlen változóban tároljuk. A következő sor már az Xdialoghoz fordul a név bekéréséhez. Az ismerős `` ` ` `` műveletjelet (operator) használjuk, ami az idézőjelek között szereplő parancsot futtatja és a kimenetét az egész kifejezés helyére helyettesíti be. Az Xdialog `--stdout` általános beállítása lehetővé teszi az alapértelmezésben a szabványos hibacsatornára érkező adatok átirányítását, így a felhasználó által begépelt adat az ablak bezárása után a szabványos kimeneti csatornán fog szerepelni, s így a `$NEVED` változó tárolja. Ebben az új Xdialog-sorban egy új általános kapcsolót láthatsz, a `--title`-t, amellyel az ablak címét írhatjuk át.
A kérdésfeltevés után meg kell vizsgálnunk, hogy milyen módon zárta be az ablakot a felhasználó. Az ***Ok*** gomb megnyomása esetén a programnak másként kell eljárnia, mint a ***Mégsem*** gomb választásakor, ezért a `$?` különleges változó értékét vizsgáljuk meg. Ez a változó a `bash`-ben az utoljára futtatott program visszatérési értékét tartalmazza. A fentebb említettek alapján választjuk szét az eseteket: normál esetben a név beírását az ***Ok*** gombra történő kattintás követi – ekkor egy üzenetdobozban köszöntjük a felhasználót. A ***Mégsem*** megnyomása esetén az udvariatlan felhasználótól névtelenül köszönünk el. Ha pedig a bal felső sarokban látható X-szel zárta be az ablakot, akkor csak a szabványos kimenetre írunk ki egy üzenetet.

Hasznos beállítások

Most csupán néhány fontosabb beállítást említenék meg, bár ezeknél jóval több létezik – mind megtalálhatod őket a kényelmesen használható HTML alapú leírásban.
A `dialog` és `cdialog` parancsfájlokban a `--backtitle <szöveg> (általános)` a beállítással a háttérre, azaz magára a terminálra lehetett fejlécet kiíratni. Grafikus felület alatt a háttérre nem illik írni, de a megfeleltethetőség érdekében a beállítást meg kellett tartani. A szöveg ezzel a beállítással jelenik meg az ablakban, majd egy elválasztójel után továbbiak követik.

```
--allow-close | --no-close (általános)
```

A `--allow-close` alapértelmezett beállítás, az ablaknak az ablakkezelőn keresztül történő bezárását teszi lehetővé. Ebben az esetben az Xdialog 255-ös visszatérési értékkel ér véget. A `--no-close` megadása esetén az Xdialog figyelmen kívül hagyja az ablakkezelő bezárt eseményeit. Megjegyzendő, hogy az ablak a töröl esemény kiváltásával így is bezárható.

```
--no-buttons (átmeneti)
```

2. lista Könyvtárfa mentése

```
#!/bin/bash

TITLE="Könyvtárfa"

MSG="
Ez a program egy könyvtárról készít egy listát.
1. lépés: válaszd ki a könyvtárat
2. lépés: válaszd ki az új adatbázist
"
Xdialog --title "$TITLE" --noclose --stdout
➥--msgbox "$MSG" 0 0

BACKTITLE="Válaszd ki a könyvtárat"
DIR=`Xdialog --backtitle "$BACKTITLE"
➥--title "$TITLE" --no-close --stdout
➥--no-buttons --dselect "" 0 0`
if [ 1 -eq $? ]; then
    exit 1
fi

f=1
while [ 1 -eq $f ]; do
    BACKTITLE="Válaszd ki a célfájlt"
    FILE=`Xdialog --backtitle "$BACKTITLE"
    ➥--title "$TITLE" --no-close --stdout
    ➥--fselect "" 0 0`
    if [ 1 -eq $? ]; then
       exit 1
    fi

    if [ -f $FILE ]; then
       Xdialog --title "$TITLE" --no-close
       ➥--stdout --default-no --yesno
       ➥"A célfájl létezik. Felülírjam?" 0 0
        if [ 0 -eq $? ]; then
           f=0
       fi
    else
       f=0
    fi
done

updatedb --localpaths="$DIR"
➥--output="$FILE"; echo "XXXX" |
Xdialog --title "$TITLE" --no-close --stdout
➥--no-buttons --infobox "Kérlek, várj,
➥amíg elkészülök..." 0 0
```

3. lista Könyvtárfa böngészése

```
#!/bin/bash

TITLE="Könyvtárfa"

BACKTITLE="Válaszd ki az adatbázist!"
FILE=`Xdialog --backtitle "$BACKTITLE"
➥--title "$TITLE" --no-close --stdout
➥--no-buttons --fselect "" 0 0`
if [ 1 -eq $? ]; then
    exit 1
fi

echo "Xdialog --title \"$TITLE\" --stdout
➥--treeview \"$FILE\" 0 0 0 \\" >/tmp/dir$$
i=2; IFS="
"
PROGRESS="Fa készítése..."
locate -d "$FILE" \* | while read LINE; do
    while [ "" == "`echo "$LINE" | cut -d/
    ➥-f$i`" ]; do
       i=`expr $i - 1`
    done
    until [ "" == "`echo "$LINE" | cut -d/
    ➥-f$i`" ]; do
       echo -n " \"`echo "$LINE" | cut -d/
       ➥-f-$i`\"" >> /tmp/dir$$
       echo -n " \"`echo "$LINE" | cut -d/
       ➥-f$i`\"" >> /tmp/dir$$
       echo " off `expr $i - 2` \\"
       ➥>> /tmp/dir$$
       i=`expr $i + 1`
    done
    echo -n "."
done | Xdialog --title "$TITLE" --progress
➥"$PROGRESS" 0 0 `locate -d "$FILE" \* |
➥wc -l`

echo " ;" >> /tmp/dir$$
. /tmp/dir$$
rm /tmp/dir$$
```

Ezzel a beállítással lehet elérni, hogy ne jelenjen meg az ***Ok*** vagy a ***Mégsem*** a `--infobox` dobozban, illetve a ***Könyvtár létrehozása***, a ***Fájl törlése*** és a ***Fájl átnevezése*** gombok a `--fselect`, `--dselect` dobozokban.

`--default-no (átmeneti)`

A ***Nem***, illetve a ***Mégsem*** gomb lesz az alapértelmezett, vagyis ami előre ki lett választva.

Fontosabb dobozok

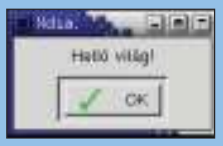
Az alábbiakban tárgyalt dobozok jóval többre képesek, mint amennyit itt elárulok róluk. Nem mutatom be az összes kapcsolójukat és helyhiány miatt nem teszek említést az összes átmeneti beállításról sem, amelyekkel használhatók.

`--yesno <szöveg> <magasság> <szélesség>`

A `--yesno` mindössze annyiban különbözik a `--msgbox`-tól, hogy egy ***Ok*** gomb helyett egy ***Igen*** és egy ***Nem*** gombot rak ki a szöveg kiírása után.

`--infobox <szöveg> <magasság> <szélesség> [<idő>]`

Hasonlít a `--msgbox`-ra, de egy, a `--no-buttons` gombok



1. kép

2. kép

3. kép

nélküli ablakhoz jutunk. Továbbá egy nem kötelezően megadandó időértékkel meghatározhatjuk, hogy hány ezredmásodpercig éljen az ablak, utána önműködően bezárul. Ha az időérték 0 vagy elhagyjuk, a doboz másként viselkedik: a szabványos bemenetről fogad üzeneteket.

```
--progress <szöveg> <magasság> <szélesség>
➥[<max pont>]
```

Ez egy olyan doboz, amelyben egy állapotjelző csík található. A szabványos bemenetről fogad üzeneteket.
A `<max pont>` által jelzett számú pontot vár és ennek megfelelően növeli a csíkot. A `<max pont>` elhagyása esetén alapértelmezésben 100-at vesz. Pontok helyett számokat is küldhetünk neki a szabványos bemeneten keresztül. Az ablak akkor zárul be, ha a szabványos bemenetre állományvége jel érkezett, vagy a pontok száma, illetve az átadott szám átlépte a `<max pont>` értékét.

```
--treeview <szöveg> <magasság> <szélesség>
➥<listahossz> <címke1> <elem1> <állapot1>
➥<mélység1> ...
```

Ez a doboz az átadott elemekből egy fát épít, amely legfeljebb 24 szint mélységű lehet. A `<címken> <elemn> <állapotn> <mélységn>` sorozat tetszőleges számban ismétlődhet. Egy elem mélysége az előzőnél legfeljebb eggyel lehet nagyobb, kisebb viszont akármennyivel.
A `<listahossz>` mondja meg, hogy hány soros legyen az az alablak, amelyben a fát ki kell rajzolni; 0 megadása esetén ez önműködő lesz. Az ***Ok*** megnyomása révén a szabványos kimeneten megjelenik a fa kijelölt elemének a címkéje.

```
--fselect <állomány> <magasság> <szélesség>
--dselect <könyvtár> <magasság> <szélesség>
```

Ezekkel a dobozokkal a GTK+ állományválasztóját érhetjük el: egy állomány- vagy könyvtárnevet választhatunk ki velük. Az `<állomány>` és a `<könyvtár>` beállítások az alapértelmezést jelölik ki. A `--no-buttons` segítségével tüntethetők el a ***Könyvtár létrehozása***, a ***Fájl törlése*** és a ***Fájl átnevezése*** gombok.

A hegycsúcs

Most látni fogod, hogy mennyire egyszerűen lehet összetett feladatokat megoldani az Xdialog használatával. Két parancsfájlt fogunk írni: az első egy tetszőleges könyvtár teljes szerkezetét menti (*2. lista*), a második képes az első által készített adatbázisokból betölteni (*3. lista*) a szerkezetet és egy fát készíteni belőle (lásd a *3. képen*).
Ha ismered a `bash`-t és figyelmesen követted az eddigieket, ezek a sorok nem jelentenek meglepetést. Az első a könyvtár és az adatbázis megkérdezése után leellenőrzi, hogy az állomány létezik-e, majd egy türelemre felszólító üzenet mellett elkészíti a könyvtárlistát. A második egy kicsit összetettebb: az adatbázis ismeretében a `locate` paranccsal az összes bejegyzést kiszedi belőle, s ezeket soronként feldolgozva egy átmeneti állományban előállítja az Xdialog-paranccssort. Ezt egy csővezetékkel oldjuk meg, az állapotjelző csík táplálásához azonban még szükség van egy csővezetékre.

Miért ne?

Eddig csak az Xdialog mellett szóló érvekkel hozakodtam elő, ezért joggal megkérdezheted, hogy mikor fogok végre hibát keresni benne. Épp az előbb tettem meg. Ekkora méretű alkalmazásokat már nem gazdaságos `bash`-ben megoldani. Véletlenül se próbálj meg például CD-katalógust készíteni a fenti programokkal, mert napokig is eltarthat, mire elkészíti a fát. Az Xdialog telepítő parancsfájlnak tökéletes megoldás. Ha fontos a csillogás, készíthetsz hajlékonylemez-formázó programot és szép grafikus kabátot húzhatsz rá. Látni kell viszont a héjprogramozás korlátait: az előbbi feladat megoldása lassú és nem is szép. Egy szint elérése után nem kerülhető el, hogy megtanuld a grafikus felhasználói felületek készítésének a módjait. Oldd meg a feladatot C++-ban, Javában, vagy – ha közel állnak hozzád a parancsnyelvek – Tcl/Tk-ban.
Remélem, sikerült bemutatnom az Xdialog nyújtotta lehetőségeket. Ha te magad, a barátnőd vagy a felettesed szereti a grafikus felületet, és egy egyszerű, parancsfájllal önműködővé tehető feladat áll előtted, az Xdialog a te barátod.

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

Fájlkezelés – másként

A kétpaneles fájlkezelőkről.

Bizonyára mindenki találkozott már a KDE vagy a Gnome beépített fájlkezelőjével, amelyben az ablak nagy részét ikonok formájában megjelenő könyvtárak és más állományok foglalják el, a bal oldalon pedig néhány adatot olvashatunk róluk. Ez hatékony lehet, ha valaki az egeret szereti használni és minden csínját-bínját kitanulja a módszernek. Létezik azonban egy másik út is…

A kétpanelesekről általában

Aki még nem látott ilyen elrendezésű fájlkezelőt, annak merőben szokatlan lehet az elv: ugyanazon az ablakon belül két egymás melletti panelen két könyvtár tartalmát jeleníti meg listanézetben. Így az állományok neve fontosabb szerepet kap, mint az ikonjuk, sőt mint látni fogjuk, nem is feltétlenül szükséges ikont rendelni hozzájuk. A név mellett alapbeállításként az összes kilistázott elemről részletes adatok jelennek meg (típusa, mérete, az utolsó módosítás dátuma). Közös tulajdonságuk még, hogy néhány egyszerű billentyűparancs elsajátításával jelentősen fel lehet gyorsítani a munkát: a panelen belül az elrendezésből következően a kurzorbillentyűkkel csak felfelé és lefelé lehet mozogni, a panelek között a TAB billentyű lenyomásával lehet váltani. Továbbá megkönnyíti a dolgunkat, ha tudjuk, hogy állományt kijelölni az INSERT billentyűvel, az egyik oldalról a másikra másolni pedig F5-tel lehet. Az F6 billentyűvel a kijelölt állományt áthelyezhetjük, az F3, illetve F4 billentyűkkel pedig belenézhetünk vagy szerkeszthetjük őket. A fontosabb parancsok gyorsbillentyűi általában a képernyő alján vannak megjelenítve, de a legtöbb esetben be tudjuk állítani magunknak, hogy melyik billentyű lenyomására mit válaszoljon a program.

1. kép
A Portos Commander első pillantásra furcsának tűnhet zebracsíkjaival

A Pcmm – a zebra

A Pcmm (teljes nevén Portos Commander) első látásra kissé furának tűnhet, ugyanis fejlesztői alapbeállításként csíkos hátteret adtak az állománylistának. Ez abból a szempontból nagyon hasznos, hogy jól elkülöníti az egymás alatt látható bejegyzéseket, viszont, ha gyorsan haladunk lefelé, a felfelé futó zebracsíkok kissé összefolynak. A név mellett szereplő kis ikonokat a KDE fájltípus-beállításai alapján rendeli az állományokhoz, és ugyanezen rendszer alapján ismeri fel, hogy milyen programmal lehet őket megnyitni. A *Név* oszlop szélességét önműködően növeli, ha olyan könyvtárba lépünk, amelyben egy nagyon hosszú nevű állomány található, viszont amint elhagyjuk az adott könyvtárat, nem csökkenti vissza a méretet. Ez azt eredményezheti, hogy bár a nevet jól látjuk, a többi tulajdonság eltűnik a látható mezőből, ezért kénytelenek vagyunk saját kezűleg csökkenteni az oszlop szélességét vagy az alsó gördítősávot használni, ha

2. kép A Krusader: korunk lovagja

3. kép
A Gnome Commander szemkímélő színekkel

kíváncsiak vagyunk például a fájlok tulajdonosaira. Ha az egér bal gombjával egy állományt áthúzunk az egyik oldalról a másikra, megkérdezi, hogy másolni vagy áthelyezni szeretnénk-e, esetleg csak egy hivatkozást hozzon létre a célhelyen, ami a forrásra mutat. Jobb gombbal kattintva kellemes kinézetű menüt kapunk, ami minden bizonnyal ismerős: lehet memóriába másolni, onnan beilleszteni, törölni, átnevezni, könyvtárat létrehozni stb. A SPACE billentyű lenyomására is kijelöli a fájlokat, de semmilyen kiegészítő adatot nem jelenít meg az adott könyvtárról vagy fájlról. A ***beállítások*** menü viszonylag kevés lehetőséget ajánl fel, a könyvjelzők, a használt szövegszerkesztő és a fájlokról megjelenített adatok megadásán kívül semmi lényegeset.

Krusader – a KDE lovagja

Mivel a Krusader a KDE-csomag része, sok tekintetben függ tőle: ikonjait és fáljtársítás-tulajdonságait teljes egészében a grafikus felület beállításaiból veszi, tulajdonképpen úgy viselkedik, mintha a Konqueror fájlkezelőt bújtatták volna más bőrbe. Jobb gombra való kattintással ugyanaz a menü látható, bal gombbal történő áthúzás esetén pedig éppúgy megkérdezi, hogy másolni, áthelyezni vagy hivatkozást létrehozni szeretnénk-e, vagy mégsem csinálunk semmit. A bal gomb egyszeri

4. kép A Midnight Commander

Az 57. CD Magazin/Fajlkereses könyvtárában megtalálható a programok forráskódja, s a következő parancsokkal tudjuk telepíteni őket:

```
tar -xzf csomag_neve.tar.gz
cd csomag_neve
./configure
make
make install
//Ehhez a lépéshez
rendszergazdai jogosultságok
szükségesek.
```

Ha a parancsok hiba nélkül lefutnak, azonnal ki is próbálhatjuk őket.

kattintására az adott fáljt kijelöli. Másolás és áthelyezés közben a már jól ismert kis ablakocska mutatja, hol tartunk. A SPACE billentyű lenyomására ugyan nem jelöli ki az állományokat, ellenben van egy nagyon hasznos tulajdonsága: ha a fájl felett nyomtuk meg a billentyűt, akkor egy kis ablakban egyszerűen kiírja a méretét; ha viszont könyvtáron álltunk, akkor a könyvtár teljes helyfoglalását kiszámolja az összes alkönyvtárával és tartalmazott állományok méretével együtt. A Krusader rendkívül sok beállítási lehetőséget ad: az ***eszközök*** menüben a Konfiguratort választva egy kellemes üdvözlőkép után a bal oldalon tudunk a különböző kategóriák közt lépkedni. Megadhatjuk, hogy milyen beállításokkal induljon el, hogyan nézzen ki a felhasználói felület, illetve hogy milyen betűtípust használjon.

Két olyan beállítási lehetőség is van, amelyekre külön is ki szeretnék térni: az egyik a fájlrendszerek önbefűzése a könyvtárfába (vagyis az `automount`). Ez olyan környezetben lehet hasznos, amelyben a rendszer nem támogatja az önbefűzést; ilyen esetekben a Krusader az adott lemezt egy ideiglenes könyvtárba fűzi be, mindezt számunkra észrevehetetlenül, majd amikor kilépünk a könyvtárból, le is választja. A Krusader másik hasznos tudása, hogy a tömörített fájlokat, mentéseket képes könyvtárként kezelni. Hasonlóan az előbbihez, ha itt belépünk egy archivált fájlba, akkor azt egy ideiglenes könyvtárba kicsomagolja, amelyben aztán szabadon mászkálhatunk, törölhetünk belőle, másolhatunk bele – majd a könyvtár elhagyásakor a kiindulási helyen találjuk magunkat az újra becsomagolt a fájllal, de immár az esetlegesen megváltoztatott tartalommal.

Gnome Commander

Miként a Krusader a KDE-vel, a Gnome Commander – mint neve is mutatja – a Gnome grafikus felülettel áll szoros kapcsolatban: az alapértelmezett programokat a grafikus felület beállításaiból nyeri, és a másolást is a kis felbukkanó ablakban követhetjük nyomon.

Az egér jobb gombjával a szokásos menüt tudjuk előcsalogatni, a bal gomb viszont csak a kurzort mozgatja, az állományokat nem jelöli ki. A SPACE billentyűt lenyomva itt is a könyvtár teljes méretét ismerhetjük meg, de nem egy különálló ablakban, hanem a ***méret*** oszlopban, amelyben eddig a `<DIR>` bejegyzés szerepelt. Amint elkezdünk valamit gépelni, az nyomban a parancssorban jelenik meg, és az ENTER lenyomására végre is hajtódik. A beállítási lehetőségek száma viszonylag nagy, külön megemlíteném a grafikus lehetőségeket: az alapértelmezett „sötét háttéren világos betűk” témát lecserélhetjük annak fordítottjára, sőt akár valamennyi összetevő színét mi magunk adhatjuk meg. Kérhetjük, hogy például a különböző típusú állományok más-más színnel jelenjenek meg. FTP-ügyfélként is képes működni, és a hálózaton megosztott erőforrásokat is el tudja Sambán keresztül érni.

mc – egy más világ

A Midnight Commander egy kissé kilóg a sorból, hiszen eddig a grafikus felületen futó alkalmazásokat vettünk sorra, ezzel szemben a Midnight Commander karakteres felületen fut. Kinézete szinte egyáltalán nem hasonlít az eddig tárgyalt három fájlkezelőre, csak az elrendezése és a működése azonos velük. Jó szolgálatot tehet azoknak, akik szeretnének megismerkedni a konzol kínálta lehetőségekkel, esetleg grafikus felület nélküli számítógépen kell dolgozniuk.

Szöveges állományok megjelenítésére és szerkesztésére saját programot használ (mcedit), a fájltársításokat a rendszer beállításaiból veszi. Kis erőforrásigényének köszönhetően nagyon gyors és hatékony. Ha valamit a parancssorba begépeltünk és a futtatása után kíváncsiak vagyunk az eredményére, nem kell minduntalan kilépni (ezt egyébként az F10 billentyűvel vagy az `exit` parancs beírásával tehetjük meg), elég megnyomni a CTRL+O billentyűkombinációt, és máris láthatóvá válik a szöveges felület. A kombináció ismételt lenyomásával vissza tudunk térni az `mc`-hez. Beállításokban itt sincs hiány: a színeket, a panelek elhelyezését, a mutatott fájltípusokat és még sok mindent mi magunk állíthatunk be.

Végezetül

Nem kívánok győztest hirdetni a kétpaneles fájlkezelők között, mivel ez nem egy teszt, másrészt mindenkit arra szeretnék bátorítani, hogy használja őket, és találja meg a neki legmegfelelőbbet.

Kellemes próbálkozást mindenkinek!

Nagy Lénárd
(nagylenard@monornet.hu)
Másodéves hallgató a BMF Neumann János informatika karán. Szabadidejét többnyire a barátaival tölti, hobbija a programozás és a néptánc.

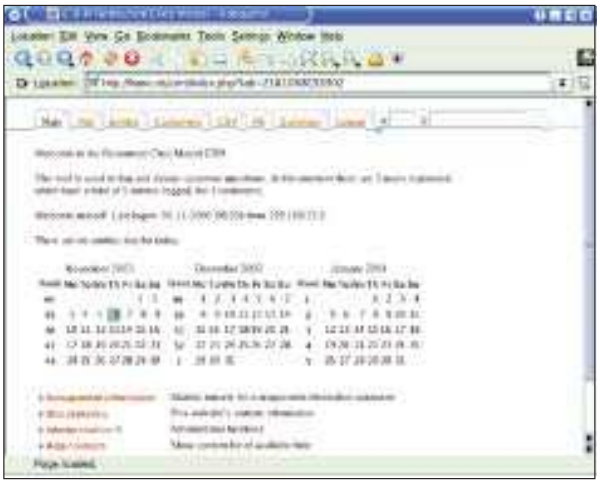
Az ügyfél mindenekelőtt

Előfordul, hogy a bor kevésnek bizonyul ahhoz, hogy üzletfeleinket boldoggá tegyük: kövessük nyomon az igényeiket, beleértve az értékesítés előkészítésével kapcsolatos tudnivalókat, a terméktámogatást és a személyes találkozókat is.

Mostani írásom témája, François, a webalapú intelligencia. A mai menü tervezgetése közben már azoknak a lehetőségeknek a puszta száma is mellbe vágott, amelyeket ez a kifejezés eszembe juttatott. Téged is, mon ami? Igaz, hogy beszélhetnénk a webalapú mesterségesintelligencia-programokról, de én valami egyszerűbbre gondoltam. Enciklopédiák, üzletek közötti készletáramlások, szótárak, keresőmotorok, sőt még az elektronikus levél is. Igen, François, osztom a véleményedet. A manapság tapasztalható rengeteg levélszemét okán az elektronikus levelezésben is megfontolás tárgyává válik az intelligencia. Mégis: az elektronikus levél a korszerű kapcsolattartás nélkülözhetetlen eszköze. Gondold csak meg, hogy az elektronikus levélhez szorosan kapcsolódva a csoportmunka-megoldások is egyre elterjedtebbek, és talán már kezded kapisgálni, hogy miért ezt választottam mai menünk témájául.

Később kell folytatnunk, François, a vendégek megérkeztek. Bonjour, mes amis, isten hozott titeket Chez Marcelnél, a Linux világának legkitűnőbb éttermében és legjobb borospincéjében! Foglaljatok helyet, helyezzétek magatokat kényelembe – François azonnal hozza a bort. A borospincébe, François, déli szárny! Valami spanyol nedű kellene nekünk: a Bierzo Corullón 2000 tökéletes választás mai menünkhöz.

Mielőtt megérkeztetek, François-val éppen a webalapú mesterséges intelligenciáról beszélgettünk. Üzleti szempontból nézve hajlok arra a véleményre, hogy a legfontosabb dolog mindig az ügyfelekkel kapcsolatos. Ügyfelek nélkül nincs üzlet, ebből következően igen nagy energiát kötnek le az ügyfélkapcsolatokat ápoló rendszerek. És itt jönnek a képbe a CRM (customer relationship management, ügyfélkapcsolat-kezelő) programok: ezek a programok mindenféle adatot összegyűjtenek az ügyfelekkel kapcsolatban, az alapvető kapcsolattartási részletektől kezdve az eladási, marketing- és támogatási adatokig. Ezeket az ismereteket felhasználva a vállalatok az ügyfelek igényeihez igazodva hozhatnak létre feladatokat és emlékeztetőket. Ez lehet egy bizonyos termékkel kapcsolatos telefonhívás, egy kérés kezelése, reklamáció intézése, egy találkozó időpontjának a megbeszélése vagy bármi más, az ügyfelekkel kapcsolatos teendő. A CRM-rendszerek a méretüket

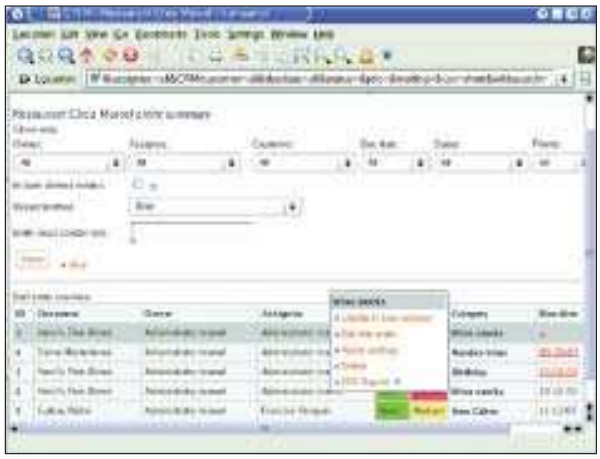


1. kép A CRM-ctt főképernyője, amely felügyeleti és beállítási lehetőségeket kínál

tekintve a nagyon egyszerűektől a rendkívül összetett, teljes céget lefedő, adatbányász behemótokig terjedhetnek. Az áruk is hasonlóan nagy szórást mutat az ingyenes vagy nagyon olcsó programoktól kezdve a többszázezer dolláros rendszerekig. Már e vonatkozás alapján is képet alkothatunk arról, hogy milyen fontosságot tulajdoníthatnak az üzleti életben az ügyfélkapcsolatoknak.

Itt, a mi kis Linux-konyhánkban a nyílt forrással dolgozó programozók segítségével belekóstolhatunk néhány CRM-rendszerbe, anélkül, hogy ez a ráfordított időtől eltekintve bármibe is kerülne.

Az első CRM-csomag, amit szemügyre szeretnék venni, a CRM customer tracking system (CRM-ctt, azaz a CRM ügyfél-nyomkövető rendszer) nevet viseli. Webes alapfelülete annyira egyszerű, hogy az embert arra késztetheti, hogy ne is foglalkozzon vele – bevallom, majdnem én is így jártam. Pedig egyszerűsége mögött egy jó képességű CRM-rendszer rejtőzik. Több felhasználó és többféle nyelv kezelését foglalja magában, továbbá gazdag paraméterezhetőséget, biztonsági beállításokat, elektronikus levélben érkező emlékeztető és PDF formátumú jelentések készítésének a lehetőségét, s ezekkel a tulajdonságaival a CRM-ctt mély benyomást



2. kép A CRM-ctt egyszerű jelentéskészítési és egy kattintásra működő PDF-export lehetőségeket teremt

gyakorol kipróbálójára (*1. kép*). A program egy példányát a ➲ http://crm-ctt.sourceforge.net címről tölthetjük le.
A rendszer üzembe helyezéséhez és futtatásához egy futó, PHP- és MySQL-támogatással rendelkező Apache-kiszolgálóra van szükségünk. Amennyiben a CRM-ctt grafikus támogatására is igényt tartunk, a `php-gd` csomagot is telepíteni kell a rendszeren. A telepítés több lépésből áll, de egyáltalán nem bonyolult: a csomagot bontsuk ki webkiszolgálónk dokumentumkönyvtárának a gyökerébe. Valószínűleg a könyvtár nevét is meg akarjuk majd változtatni:

```
cd /var/www/html
tar -xzvf crm-1.9.2-19102003.tar.gz
mv crm-1.9.2-19102003 crm
```

Mielőtt folytatnánk, a CRM könyvtár tulajdonosának állítsuk be webkiszolgálónk felhasználónevét és csoportját. Ez az én esetemben a `chown -R apache.apache crm` parancs kiadását jelentette.
Folytassuk a program beállításaival! Kezdjük azzal, hogy böngészőprogramunkat arra a helyre irányítjuk, ahova a programot telepítettük (olyasmit kell címként megadnunk, mint ➲ http://www.webkiszolgalo.dom/crm). Ezután a program egy csomó beállítóképernyőn vezet végig bennünket, ezek közül utolsóként a vállalat adattárházát kell létrehoznunk. Kövessük tovább a lépéseket: adjuk meg a nevünket, a rendszergazda nevét és levélcímét, valamint a kiválasztott rendszergazdai jelszót. Mindössze négy webalapú lépésről van szó.
A telepítési folyamat befejező lépése a beállítások lemezre íratása. A javasolt megoldás erre a ***header.inc.php*** fájl jogosultságainak ideiglenes megváltoztatása:

```
cd /var/www/html/crm
chmod 777 header.in.php
```

Térjünk vissza a telepítési oldalra és kattintsunk a ***Try Now*** (kipróbálás) feliratra. Ha minden jól megy, üzenetet kell kapnunk a sikeres befejezésről. Választhatjuk a fájl előállításának és a CRM telepítési könyvtárába való kézzel bemásolásának módját is. Ha ezen a lépésen túlvagyunk, a ***header.in.php*** engedélyét a 777-es értékről váltsuk vissza 755-re. Most menjünk az oldal aljára: kattintsunk a ***When done, point your browser here*** (Amikor kész, irányítsd ide a böngésződet) kiemelt feliratra, ami a beléptetőképernyőre visz bennünket. A belépéshez a rendszergazdai nevünket és jelszavunkat használhatjuk.
Egy ilyen rendszert nem igazán tudunk ügyféladatok nélkül használni, úgyhogy lássunk is hozzá a beírásukhoz. Kattintsunk a ***Customer*** (ügyfél) fülre, majd a megjelenő oldalon kattintsunk az ***Add a customer*** (ügyfél felvétele) feliratra. Ismételjük meg a műveletet az összes ügyfelünkre vonatkozóan! Ha ügyfeleink adatainak a felvitelével készen vagyunk, akkor már ügyeket (***Entities***) is létrehozhatunk hozzájuk. Egy ügy bármi lehet, amit az adott ügyféllel kapcsolatban tennünk kell: egy folyamatban lévő rendelés, egy hibajegy (trouble ticket) megnyitása, egy barátságos emlékeztető telefonhívás vagy virágküldés valakinek a születésnapjára. A CRM-ctt még elektronikus levél küldésével is képes emlékeztetni a felhasználót ezekre a teendőkre. A CRM-ctt felfedezése során megismerhetjük a program nagyszerű jelentéskészítő képességeit is, amellyel többek között profi színvonalú jelentéseinket exportálhatjuk egy kattintással PDF-állományba.

3. kép Az eGroupWare kínálatában naptárt, hibajegykezelő rendszert és felhasználói webhelykezelést is találhatunk

4. kép Egy hibajegy megkezdése az eGroupWare-ben

Mivel a CRM képes rá, hogy az alkalmazottakat figyelmeztesse arra, hogy kivel mikor kell találkozniuk, fel kell vennünk egy bejegyzést a cron-ba is, hogy ezeket az elektronikus levélben érkező emlékeztetőket ki tudja küldeni. Ennek az az érdekessége, hogy a leveleket küldő kiszolgálónak nem feltétlenül kell a CRM-ctt programot futtató gépnek lennie. Én most csak egyetlen bejegyzést fogok mutatni (reggel 8 órára), de a figyelmeztető eljárást annyiszor futtathatjuk, ahányszor a cégünk szempontjából szükségesnek látjuk. A következő példában látható ***yourCRONpwd*** jelszó a CRM-ctt főoldalán lévő ***Administration*** (felügyelet) menüjében állítható be, a ***Change global system's values*** (az általános rendszerértékek megváltoztatása) menüpontban:

```
# CRM Alarm date manager
0 8 * * * wget http://www.webkiszolgalo.dom/crm/
➥duedate-notify-cron.php?password=yourCRONpwd\
➥&reposnr=XXX 1> /dev/null 2> /dev/null
```

Rendszergazdaként további felhasználói fiókok létrehozására van lehetőségünk, ilyen módon cégünk más tagjait is megbízhatjuk az egyes feladatok végrehajtásával. A felügyeleti képernyőről a teljes ügyféladatbázisra vonatkozó vezetői jelentések létrehozására nyílik lehetőségünk, amelyek szintén exportálhatók PDF formátumú jelentésekként. Mindezek végrehajtásához a CRM-ctt hálózaton keresztül elérhető kezelői leírást kínál.

Számos CRM-csomag futtatható Linuxon, s ezek mindegyikére jellemző, hogy a tevékenységük középpontjában az ügyféllel való kapcsolattartás nyomon követése áll. Közelebbről megvizsgálva mégis az az érzésünk támadhat, hogy a CRM-csomagok nagymértékben hasonlítanak a korszerű webalapú csoportmunkát támogató programcsomagokra. És nem is gondoljuk rosszul. Egy jól működő CRM-rendszer számos eleme megtalálható egy jó csoportmunka-programban is. Ami a csoportmunkaprogramot mégis megkülönbözteti, az az általa nyújtott webalapú intelligencia területe és az azáltal megnyíló, együttműködést támogató lehetőségek. A csoportmunka-programcsomagok központi elektronikus levelezéssel, naptárakkal, címjegyzékekkel, az eszmecseréknek teret adó fórumokkal, hívás-nyomonkövetéssel, és még számos egyéb programmal rendelkezhetnek. Az egyik ilyen csoportmunkacsomag, amely igen érdemes a vizsgálatra, az eGroupWare.

Az eGroupWare nyílt forráskódú API-val rendelkező GPL-programcsomag, így könnyen hozhatunk létre és adhatunk további programokat a rendszerhez. A programok között elektronikus levelezőrendszert, naptárt, naplózó rendszert az ügyfélhívások és teendők nyilvántartására, hibajegyrendszert (trouble ticket system), személyes és szervezeti telefonkönyvet, tudásbázist, egy wiki-dokumentációs rendszert, és még sok egyebet találhatunk. Ezenkívül webhelykezelő rendszert is tartalmaz, amellyel az egyes felhasználók elkészíthetik a saját oldalaikat. Mindezekre az eGroupWare képes, miközben takaros, munkára kész felületet is a magáénak mondhat (*3. kép*). Írásom születésekor (2003 novemberének az elején) az eGroupWare-t körülbelül két hét választja el 1.0-s változatának megjelenésétől (sajnos még mindig elválasztja ettől az a néhány hét, a jelenlegi változat az eGroupWare 1.0 RC3.) Az eGroupWare támogatja a legelterjedtebb nyílt forrású adatbázis-formátumokat, a MySQL-t és a PostgreSQL-t. A kipróbáláshoz a PostgreSQL-t választottam, de bármelyik könnyen beállítható. Ha be szeretnénk szerezni az eGroupWare egy példányát, a csomagot megtalálhatjuk az 57. CD Magazin/Fogado könyvtárában. Ezen az oldalon remélhetőleg megtalálhatók a mindenki számára megfelelő csomagok. A csomagot a ***tar.bz2*** fájlból és először bontsuk ki webkiszolgálónk dokumentumkönyvtárának a gyökerébe. Például egy alapértelmezett Apache-telepítés esetén ennek a könyvtárnak a */usr/local/apache/htdocs* elérési útvonalon kell lennie, de egy RPM-alapú telepítésnél (mint amilyen a Red Hat) ez gyakran a következő:

```
/var/www/html könyvtár:
cd /var/www/html
tar -xzvf eGroupWare-változatszám.tar.gz
```

Ez a lépés egy ***egroupware*** nevű könyvtárat hoz létre. A könyvtár engedélyeit változtassuk meg, hogy a webkiszolgálónk jogaival teljes körűen hozzáférjünk; ez nálam a következőt kjelentette:

```
chown -R apache.apache egroupware
```

Feltéve, hogy PostgreSQL-t szeretnénk hozzá használni, a következő lépés egy PostgreSQL-felhasználó létrehozása az adatbázis eléréséhez. Ezt a postgres felhasználónkra való váltással tehetjük meg:

```
$ su - postgres
$ createuser egroupware
Shall the new user be allowed to create
databases? (y/n) y
Shall the new user be allowed to create more new
users? (y/n) n
CREATE USER
```

Amikor a ***Shall the new user be allowed to create databases? (y/n)*** (Létrehozhat-e az új felhasználó adatbázisokat?) kérdésre kell felelnünk, adjunk igenlő (y) választ. Amikor azt kell megadnunk, hogy az új felhasználó létrehozhat-e új felhasználókat (***Shall the new user be allowed to create more new users?***), válasszuk az „n" (nem) lehetőséget. Már csak egyetlen teendőnk maradt: hozzuk létre az adatbázist az eGroupWare számára. Még mindig a postgres felhasználóval bejelentkezve írjuk be a következő parancsot:

```
$ createdb -U egroupware egroupware_db
CREATE DATABASE
```

Ha az adatbázist nem szeretnénk egroupware-nek nevezni, tulajdonképpen akárminek hívhatjuk vagy egy kicsit meg is változtathatjuk a nevét, ahogy a fenti példában én is tettem. Ezzel parancssoros teendőinkkel készen is vagyunk, izzítsuk be a böngészőprogramot és véglegesítsük az eGroupWare beállításait. Indítsuk el a Mozillát, a Konquerort vagy bármilyen más, Java-parancsfájlok futtatásra képes böngészőt, és webkiszolgálónkat irányítsuk a ➲ http://webkiszolgalonk/egroupware/setup könyvtárra.

A beállításokat tartalmazó fejállomány (***header.inc.php***) létrehozásához ezen a képernyőn írjuk be a beállításainknak megfelelő adatokat. Ha adatbázis-felhasználónknak nem az egroupware nevet adtuk, ne feledjük el itt azonosítani. Ugyanez vonatkozik az adatbázis nevére is. Itt kell beállítanunk a fejállományra (***header.inc.php***) vonatkozó jelszót és a rendszerfelügyelői jelszót. A fejállomány jelszavával módosíthatjuk vagy hozhatjuk újra létre a most létrejövő ***header.inc.php*** fájlt.
Amikor ezzel is készen vagyunk, a ***setup/header*** (beállítások/fejállomány) belépési képernyőjére jutunk. Egyszer már létrehoztuk a fejállományt, úgyhogy valószínűleg nem akarjuk újrakezdeni, non? Most tehát a pillanatnyi eGroupWare-beállítás érdekel bennünket. De mielőtt megtennénk ezt a lépést – úgy látom, sokan közületek üres pohár mellett üldögélnek. François, ha volnál kedves újratölteni; merci!
Amikor a rendszergazdai jelszóval beléptünk, a beállítóprogram ellenőrzi, hogy az adatbázis rendben létrejött-e és hogy a fejállomány létrehozásának lépésekor megadott felhasználói azonosítót használjuk-e. Ha eddig a pontig minden rendesen zajlott, a helyi beállítások első lépésénél (Step 1) kell tartanunk. Kattintsunk az ***Install*** (telepítés) gombra a program tábláinak létrehozásához és az eGroupWare programcsomag alkotórészeinek telepítéséhez.
A rendszer néhány percig elrágódik majd ezen a feladaton. Amikor minden készen van, ellenőrizzük a böngésző ablakát, hogy nem kaptunk-e valamilyen hibaüzenetet, majd kattintsunk a ***Recheck my Installation*** (a telepítés ellenőrzése) gombra. Ha minden normálisan zajlott, továbbléphetünk a második lépésre (Step 2), és létrehozhatjuk ***admin*** felhasználói fiókunkat. Három próbafelhasználó létrehozására is lehetőségünk nyílik, de ez nem kötelező. A harmadik lépésben (Step 3) a használni kívánt nyelvet adhatjuk meg, míg a negyedikben (Step 4) az egyes programok saját beállításait végezhetjük el. Ebben a párbeszédablakban dönthetjük el azt is, hogy az összes programot telepíteni szeretnénk-e (ez az alapértelmezett lehetőség) vagy válogatni kívánunk közülük. Amikor innen kilépünk, készen vagyunk a telepítéssel.
Itt az ideje, hogy kezdjünk is valamit az eGroupWare-rel – ehhez legelőször is be kell lépnünk az ***admin*** felhasználóval. Az a legvalószínűbb, hogy böngészőnket a ➲ http://sajat.kiszolgalonk.dom/egroupware címre irányítjuk. A képernyő tetején végig ikonokat láthatunk, amelyek az egyes csoportmunka-szolgáltatásokat jelölik. A bal oldalon a pillanatnyi programnak megfelelő menük helyezkednek el, de mindig látható egy kisebb menü a ***Home*** (saját hely), ***Preferences*** (beállítások), ***About*** (névjegy) és ***Logout*** (kijelentkezés) menüpontokkal. A kinézet és működés a saját ízlésünknek megfelelően módosítható a ***Preferences*** (beállítások) menüpontra kattintva és ott a megfelelő változtatásokat végrehajtva.

Amennyiben felügyeleti jogosultsággal rendelkezünk, teljes cégre vonatkozó változtatásokat is végrehajthatunk. Sőt arra is lehetőségünk nyílik, hogy a felhasználók által nem módosítható alapértelmezett beállításokat adjunk meg – ez nagyon hasznos lehet egy egész cég felügyeletét ellátó rendszergazda számára. A felhasználói fiókok létrehozásakor előre meghatározott programokat rendelhetünk hozzájuk, s ezzel egy csoportokon alapuló belépési rendszer alakul ki; például a támogatói csoportnak valószínűleg a hibajegyeket kezelő rendszerre lesz szüksége (*4. kép*).
Ez a csoportszemléletre épülő megközelítés átgondolt eszközkészletet kínál felhasználóink számára. Először hozzuk létre a csoportokat, döntsük el, hogy mely programok használatára lesz szükségük, s ezután ezekre a csoportokra építve hozzuk létre felhasználóinkat.
Az eGroupWare jövőjének és folyamatos támogatásának tekintetében elmondható, hogy a programcsomaghoz pezsgő fejlesztői és felhasználói közösséggel bír, számos levelezőlista és IRC-csatorna érhető el arra az esetre, ha felmerülő kérdéseinkre válaszokat keresnénk.
Mon Dieu! Már megint kifutottunk az időből. Sajnos a hely és az idő nem engedi, hogy mást is részletesen megvizsgáljunk, pedig nagyon sok kitűnő programcsomag létezik még, amelyek érdemesek lennének a figyelmünkre. Habár mindnyájan szeretünk a Linuxszal főzni, szerény meglátásom mégis az, hogy a mi vendég–vendéglátói viszonyunk nem igényel számítógépes programot. Ehelyett inkább kényelmes székeitekkel és kedvenc asztalaitokkal várlak titeket, François pedig ügyel arra, hogy a poharatok mindig tele legyen. Néha az egyszerűség a legfőbb erény. François, ha volnál szíves gondját viselni a vendégeinknek! A következő alkalomig, mes amis, ürítsük poharainkat egymás egészségére! A vôtre santé! Bon apétit!

Linux Journal 2004. február, 118. szám

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

KAPCSOLÓDÓ CÍMEK

CRM-ctt ➲ http://crm-ctt.sourceforge.net
eGroupWare ➲ http://www.egroupware.org
Marcel borlapja
➲ http://www.marcelgagne.com/wine.html

A szinte tökéletes játék: az Unreal

Anno 1996-ban a játékosok körében felröppent a hír, hogy olyan játék készül, ami új fejezetet nyit a számítógépes játékok történetében. Egészen pontosan olyan grafikája lesz, amilyet még soha senki nem látott.

Azután 1997-ben elkezdtek szállingózni a képernyőképek és a fejlesztői leírások, végül egyszer csak minden ilyesmi megszűnt. Mindenki úgy könyvelte el, hogy ez is csak egy olyan kezdeményezés volt, ami valahol az íróasztalfiókban végzi, hiszen a technika akkori szintjén a feladat megvalósíthatatlannak tűnt.
1998 volt az az év, amikor bombaként robbant a hír: megjelent az Unreal.
A szaksajtó azonnal többoldalas leírásokat, elemzéseket szentelt neki, és egybehangzóan hihetetlennek és lenyűgözőnek vélte.
Ekkor látott napvilágot az a játékmotor, amelynek igazán komoly vetélytársa csak évekkel később született meg, és amelynek képességeit a Quake3-ban – szinte elhűlve – figyeltük.

Maga a játék

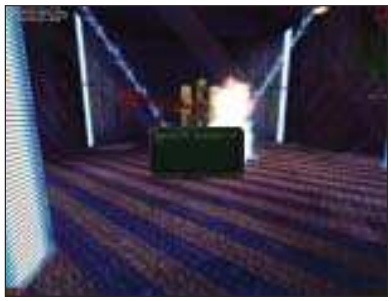
A játék lelke a világ első Unreal-motorja, amely a maga idejében forradalmi volt. Világosan látszik, hogy ennek a fejlesztők nagyon is tudatában voltak, és olyan játékot készítettek, amely „TechDemo” jelleget kölcsönöz neki. Egyaránt találkozunk HighTech-környezettel és ősrégi, komor kastéllyal, labirintussal, katedrálissal, valamint földönkívüli látvánnyal. Mindez lenyűgöző fénykezelési eljárásokkal van megtűzdelve, ami miatt bizony elvakíthat bennünket a nap, vagy éppen félhomályba borul a táj, mert a nap elé egy felhő úszik. Ebben a TechDemo-hangulatban a történet szinte elsikkad (ami szerint egy rabszállító űrhajó zuhan le a bolygóra, és az egyetlen feladat: elszabadulni innen) – még bevezető animáció sincsen, mindössze egy ősi kastély körül lebeg egy kamera.
A játék meglehetősen hosszúra sikeredett, ami előnyére vált, bár a vége felé egy kicsit egyhangú kezd lenni a játékmenet – mindezt feledteti azonban a hangulat. A játékban teljesen kidolgozott élővilágot látunk: madarak szállnak az égen, halak úsznak a vizekben, a szárazföldön pedig kisebb-nagyobb élőlények legelésznek. Ellenfeleink még ennél is változatosabbak: több méretben és felszereltségben találkozhatunk Ska-Arj harcosokkal, akik nem restek átküldeni bennünket az örök vadászmezőkre, ha tehetik. Ugyanakkor – a hasonló kategóriájú játékoktól eltérően – nem mindenki ellenség. A négykezű, békés Nalik kifejezetten barátságosak, és ha megmentjük valamelyiküket, általában mutatnak nekünk egy titkos zugot, ahol igen combos tárgyakat lehet felvenni. Érdemes tehát a védelmünkbe venni őket.

Grafika, hangok, hangulat

Ezek azok a dolgok, amelyekre egyszerűen nem lehet panasz. A látvány jelenleg már nem képviseli a technikai csúcsot, de az elsőrangú LevelDesign olyan hangulatot teremt, amelyben ezt észre sem vesszük. (Igazából ez már nem az eredeti motor, de erre a későbbiek folyamán kitérek.) A képek magukért beszélnek… A zene fantasztikus módon részét alkotja a játéknak – én a mai napig nem hallottam ennél jobb zenei aláfestést. Természetesen – akkoriban úttörő módon – dinamikusan alkalmazkodik az eseményekhez, tehát békés mászkálás vagy harc közben más-más zenét hallunk.

Néhány technikai apróság

Fontos dolog megemlíteni, hogy ez sehogyan sem lesz az „eredeti" Unreal. Ugyanis furcsa – de mégis okos – módon egy előtelepített Unreal Tournamentre van szükség a futtatásához (ez meg is látszik a grafikán). Ezzel ebben a régebbi játékban is némi technológiai fejlődéshez jutunk. Figyelemünkre méltó viszont egy, a telepítőcsomaghoz való bővítmény. Ez pedig nem más, mint az ACID-meghajtó, amely nagy sebességet tesz lehetővé a régebbi videokártyákon. Ezzel 1024×768-as felbontást is el tudunk érni egy régebbi Voodoo2 vagy TNT kártyán, ugyanakkor kihangsúlyoznám, hogy az újabb kártyákon csak ront a helyzeten. Sokat tanakodtam, miért az történik, hogy úgy hul-



lámzik tőle a kép, mintha víztükrön keresztül néznék. Lehet, hogy a beállításaimmal nincs rendben valami, vagy elképzelhető, hogy ez a meghajtó nem nVidia alapú kártyákhoz készült (és ezért nem szereti a Geforce2 kártyámat), vagy egyszerűen csak „túl gyors" lesz, azaz túlpörög. Én az utóbbira tippelek. A meghajtó nagyon jó és kegyetlenül gyors. Sajnos a Readme egy kicsit szegényes ez ügyben és nem is túl közlékeny; mindössze annyit tartalmaz, hogy mit és hol kell átírnunk (erre még a telepítés után visszatérünk).

A játék telepítése

Mint már említettem, nagyon fontos, hogy még az Unreal telepítése előtt legyen egy működőképes Unreal Tournament-példányunk (nem a 2003!). Ha ez megvan, akkor mindössze az `Unreal1` csomagra lesz szükségünk, amit a ➲ http://www.icculus.org oldalról le lehet tölteni. Látni fogjuk, hogy itt található még a küldetéslemez „Return to the Na Pali" és „Unreal-Gold" telepítője is.

A telepítés pontosan ugyanúgy zajlik, mint azt az Unreal Tournament esetében már megszokhattuk, tehát a `supermount` rendszert ki kell kapcsolni és az Unreal CD-t kézzel kell befűzni. Sajnos a szokásos Loki-telepítőről van szó, tehát erre itt is oda kell figyelni, ugyanis csak ebben az esetben látja a telepítő a forrás-CD-t. Ha a szokásos telepítési lépéseken túl vagyunk, érdemes megjegyezni a következőket: a telepítő a */usr/local/games* könyvtárban egy ***unreal*** nevű könyvtárat hoz létre, mint azt az eddigiekben megszokhattuk. A CD-ről telepíti a zenéket, a textúrákat és egyebeket. A játék motorjára és annak beállításaira azonban csak hivatkozásokat hoz létre a ***Tournament*** könyvtárból. Sőt a rejtett könyvtárként megjelenő */loki/unreal* könyvtárban (ezt a */home* alatt keressük) a ***tournament.ini*** fájljával találkozunk, amit bátran szerkeszthetünk. Ennek eredményeképpen az ***unreal*** indításakor az UT menürendszerével találkozunk, ugyanis a helyzet az, hogy valójában az UT indul és az Unreal modként (modification) fut benne. Ennek hatására az Unreal grafikája valamivel szebb lesz, ugyanakkor az lesz a dolog hátránya, hogy a gépigény a Tournament gépigényével egyezik meg. Az Unreal fantasztikus játék, kipróbálása mindenki számára kötelező, aki számítógépes játékokkal múlatja az idejét. Ez a játék is etalon lett a későbbiekben, mint ahogyan a Quake volt annak idején vagy a Half-Life lett az elkövetkező időkben. Aki még sosem próbálta, bátran vágjon bele, mert nem mindennapi élménnyel gazdagodhat; aki pedig már ismeri, ezúttal Linuxon is nosztalgiázhat. Azokban a régi szép időkben még a játék adta el a játékot, és nem a marketing. Habár Deathmatchben gyengécske (arra ott van a Tournament), az Unreal fő ereje az egyjátékos mód.

További jó szórakozást kívánok hozzá!

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

Cycling

A Cycling a kerékpárosedzést segíti: nyilvántartja a megtett körök számát, a legnagyobb sebességet, a távolságot és az egyéb adatokat. Noha eredetileg kerékpározáshoz tervezték, bátran használható futáshoz, úszáshoz vagy bármilyen sporthoz, ahol rögzíteni akarod az időt, a sebességet és a távolságot. Futtatásához szükséges: libgtk-x11-2.0, libatk-1.0, libgdk-x11-2.0, libgdk_pixbuf-2.0, libm, libpangoxft-1.0, libpangox-1.0, libpango-1.0, libgobject-2.0, libgmodule-2.0, libdl, libglib-2.0, glibc, libXi, libXext, libXft, libXrender, libfontconfig, libX11, libfreetype, libz és libexpat.

➲ http://juvenis.dyndns.org/index.php?dirid=1&artid=3

bandwidthd

Az egyetlen ok, ami miatt a bandwidthd az MRTG vagy a Cacti helyett ajánlható, az az, hogy a bandwidthd az egyedi IP-címeket ábrázolja, nem hálózati adatokat rögzít. Továbbá gazdagépenként megpróbálja osztályozni a TCP, UDP, ICMP, VPN, HTTP vagy P2P típusú forgalmat, vagyis láthatóvá

válik, hogy az egyes gazdagépek milyen forgalmat bonyolítanak az általuk használt sávszélességen. Előfeltételei: libgd, libpng12, libpcap, glibc, libXpm, libX11, libjpeg, libfreetype, libz, libm, valamint a libdl.

➲ http://bandwidthd.sourceforge.net

gtick

Metronómra van szükséged? Ez a metronóm nemcsak egyetlen zenei ütemet ad ám, hanem választhatsz, hogy 2/4-es, 3/4-es vagy 4/4-es, sőt akár egyedileg beállított ütemet adjon-e. A ritmus és a hangerő tolókapcsolóval szabályozható, amely a nyers hangolást teszi lehetővé, valamint a finomhangolás érdekében tekerőgombbal is beállítható. Futtatás: libgtk-x11-2.0, libgdk-x11-2.0, libatk-1.0, libgdk_pixbuf-2.0, libm, libpangoxft-1.0, libpangox-1.0,

libpango-1.0, libgobject-2.0, libgmodule-2.0, libdl, libgthread-2.0, libglib-2.0, libpthread, glibc, libX11, libXi, libXext, libXft, libXrender, libfontconfig, libfreetype, libz és libexpat.

➲ http://www.antcom.de/gtick

The Battle for Wesnoth

Ez a játék egy háborús játék és egy szerepjáték kereszteződése, amelyben a karakterek hatszög alakú területegységeken mozgathatók. A lények nem

különösebben erősek, de nem egyszerű eljutni a második pályára. A túléléshez találékonyságra és minden erőforrásodra szükséged lesz. Vigyázat, több órán át foglyul ejthet! Futtatásához szükségesek: libstdc++, libSDL-1.2, libpthread, glibc, libm, libgcc_s, libX11, libXext és libdl.

➲ http://www.wesnoth.org

SQL-Ledger

Azt hiszem, nem láttam még olyan programot, ami olyan gyorsan és olyan mértékben fejlődött volna, mint az SQL-Ledger az elmúlt három

évben, amióta használom. Ez a program teljes körű kettős könyvviteli szolgáltatást nyújt szinte bármilyen vállalkozás számára, méghozzá számos nyelven. Aligha találnék ennél jobb könyvelési csomagot. Rendkívül egyszerűen telepíthető. Ha a gépen már telepítve van a LaTeX, a PDF-fájlokat közvetlenül kinyomtathatod, vagy elektronikus üzenetként küldheted el a számlákat és a bevallásokat. Az alapértelmezett beállítások a kanadai adórendszerhez igazodnak, de bármilyen más rendszernek megfelelően egyszerűen átállíthatók. Az egyik legnagyszerűbb szolgáltatása, hogy az adatbázis PostgreSQL-t használ. Ha teljes könyvelési csomagot keresel a vállalkozásod számára, ezennel megtaláltad. Futtatásához szükséges: Perl, Perl-modulok – DBI, DBD (DBD-Pg vagy DBD-Oracle), PostgreSQL vagy Oracle, Perl-parancsfájlok futtatására alkalmas webkiszolgáló, webböngésző és LaTeX (nem kötelező).

➲ http://www.sql-ledger.org

Linux Journal 2004, 118. 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.

Linux bevetés közben

Tavaly év végén egy olyan könyv jelent meg, amely a rendszergazdákat és rendszermérnököket célozza meg.

Jelent már meg hasonló témájú kiadvány, ez idáig azonban nem volt szerencsém olyan könyvet a kezembe venni, amely szakmai megközelítésként azt a fajta barkácsolást választotta volna, ami a Linux-rendszereknek olyannyira a sajátja.

Tuningoljunk!

A „barkácsoljunk egy kicsit” stílus sokak számára ijesztő értelemmel bír, jelen esetben viszont mindenképpen rokonszenves a jelentése: kiszolgálóépítést és tuningolást takar. Itt a hangsúly a tuningoláson van, ami egy kicsit tudathasadássá teszi az egészet, hiszen örökös harc folyik a programozók és a rendszergazdák között a tekintetben, hogy a programozók az összes programból mindig a legújabbat szeretnék a kiszolgálógépen látni, a rendszergazdák pedig általában ragaszkodni szoktak a már jól bevált és üzembiztos (stable) változatokhoz, amelyekhez biztonsági követés, valamint jó leírás is a rendelkezésükre áll. Az író megpróbálja egyensúlyban tartani a két ellentétes igényt, azaz: építsünk olyan kiszolgálókat, amelyek biztonságosak és „friss” programokat tartalmaznak.
A tanácsok és ötletek a rendszer elindításától egészen a DNS üzemeltetéséig terjednek.

Rugalmas SSH

Először találkoztam olyan könyvvel, amelyik az SSH hihetetlenül rugalmas lehetőségeinek a teljes kimazsolázását olyan módon teszi, hogy akár egy középhaladó szakember is megérti, hogy mit és miért érdemes használnia. De akár az alagút-, a VPN- és a jail-

Adatok

Cím: Linux bevetés közben
Szerző: Rob Flickenger
Kiadás időpontja: 2003. december
Kiadó: Kiskapu Kft.
ISBN: 963 9301 55 8
Ár: 3500 Ft

építés mesterfogásait is elsajátíthatjuk, vagy a szerző útmutatásai alapján magunk is ügyes kis héjprogramokat írhatunk. Tehát minden linuxos korosztály számára ajánlható (akárcsak az Ablak–zsiráf).

Nagy segítség

A könyv első olvasásakor arra gondoltam, hogy számomra már nem tartogathat szakmai újdonságokat, azonban hamar rá kellett jönnöm, hogy a szerző olyan parancsokat ismertet meg velem, illetve olyan fajta kombinálhatóságukat mutatja be, amelyek azóta is nap mint nap megkönnyítik rendszerfelügyeleti munkámat.
A kiadvány a következő témaköröket dolgozza fel:

- a kiszolgálóépítés alapjai,
- változatkövetés,
- biztonsági mentés,
- hálózatkezelés,
- rendszerfelügyelet,
- az SSH,
- a parancsfájlok,
- adatbázis-kiszolgálók.

A könyv vége felé, amikor a DNS és az SSH beállításával ismerkedhetünk meg, a szerző egy kicsit szabadon fogja fel a biztonság kérdéskörét, a lektori megjegyzésekben azonban mindenhol szerepel jó tanácsként, hogy milyen biztonsági megfontolásokat érdemes betartani. Egyszóval ez egy igen átfogó könyv, amely teljesen rendhagyó módon közelíti meg az üzemeltetés fogalmát.
A könyv oldala a Kiskapu Kiadó honlapján a
➲ http://kiado.kiskapu.hu/51 címen érhető el. Itt megtalálható a tartalomjegyzék, letölthetők a mintaoldalak és a könyvben szereplő kódok.

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