Nyíltan vagy zártan?

A programok fejlesztése rendkívüli iramban gyorsul: gombamód szaporodnak a jobbnál jobb programok, rengeteg bővítmény készül minden operációs rendszerhez. Azt hiszem, mindezt a Linux és más nyílt forrású rendszerek térnyerésének köszönhetjük. A forráskód hozzáférhetősége nagyon sok embert arra ösztönöz, hogy saját „gondolatkísérleteiket” is közzétegyék benne, ezzel is javítva a program minőségét, és esetleg bővítve a felhasználási területét. Egyes programokra sokan azt mondják, hogy „halvaszületettek”, a nyílt forrású világban azonban nem egyszer láthattuk, hogyan is támad fel halottaiból, ha nem is azonos néven és azonos feladattal, de a forráskód szintjén mindenképpen hasznosan egy-egy ilyen projekt! Ha nincs a Linux, valószínűleg senki nem kezdett volna bele az OpenBeOS fejlesztésébe; a Linux sikerei ösztönzően hatnak az emberekre, a BeOS-rajongók pedig nem tétlenkednek, úgy döntöttek, hogyha a kedvenc rendszerük fejlesztése megszűnik, készítenek maguknak egy nyílt forrásút. Gondoljunk csak bele, hol tartanánk ma a nyílt forráskód nélkül? A Microsoft operációs rendszerei teljességgel eluralkodtak volna kicsiny bolygónkon, mindenki az önző harácsoló programírási gyakorlatnak behódolva alkotna nap mint nap, és bizony kemény pénzeket kellene leszurkolni minden egyes programért. Azt hiszem, a Linux mindenki számára hatalmas nyereség. A mostani programozópalánták ebbe a szabad világba születnek, így számukra már természetes lesz a forráskódok megosztása (hacsak el nem szegődnek a sötét oldalra). E szabadság egyik nagy kérdése és veszélye a töredezettség. Hogy mire is gondolok: Linux-változatból nagyon sok van, a ➲ http://linuxlinks.com adatai szerint a Linuxnak 285 változata létezik – ezek vagy egyedi területet fednek le, vagy „csak” egyszerű Linux-kiadások. Ez egyfelől jó, mivel így nagyon sok tapasztalatot lehet összegyűjteni, de elbizonytalanítja az embereket. Az első nagy kérdés, amit minden Linuxra vágyó felhasználó feltesz: melyiket is válasszam? Érdemes egy sokak által használt változatot beszerezni, mivel így valószínűleg sokkal hamarabb fogunk segítséget kapni, mintha egy szinte teljességgel ismeretlen Linuxot használnánk. A másik szempont az lehet, hogy közeli ismerősünk melyik változatot használja, mivel ha mi is ugyanazt használjuk, legalább a kezdeti nehézségeken át tud bennünket egy olyan ember segíteni, aki ismer minket, és ez bizony a dolgok elmagyarázása során aranyat érhet. A jövő mindenképpen a nyílt forrásé, remélem, ezt mihamarabb mindenki a magáénak fogja érezni, mert ennek a fejlődésnek semmi nem állhatja útját.

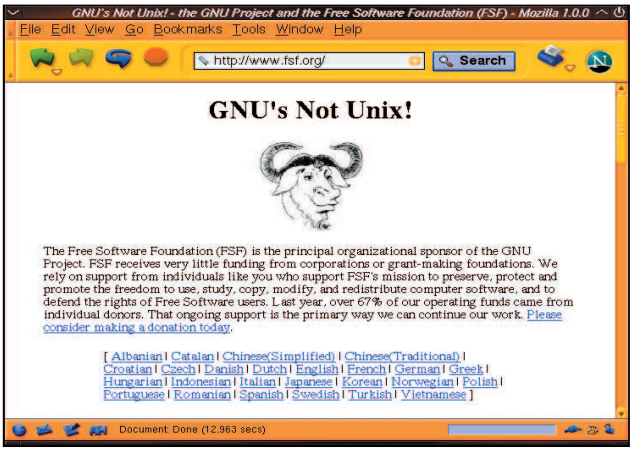
E heti nyertes linuxos viccünk:

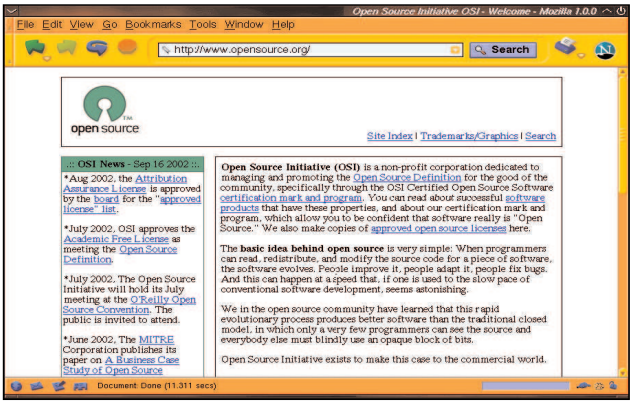
Gyorstalpaló Jedi-képzés debianosok számára:

```
#apt-get install fenykard
```

Ha a sötét oldalhoz szeretnénk tartozni, a következőképpen változik a parancs:

```
#apt-get install -d fenykard
#dpkg -i --dark-force
➥/var/cache/apt/archives/fenykard.deb
```





Mint azt a Programvadászat oldalon is írom, kérek minden olvasót, hogyha valamit a CD-mellékleten szeretne látni, írja meg a ***cd@linuxvilag.hu*** címre a Programvadászatban megadott formai követelményeknek megfelelően.
Nos, ennek fényében (fénykardjában) biztosíthatok mindenkit, hogy senki sem fog unatkozni a Linux háza táján, és reméljük, valaki tényleg elkészíti a fénykardcsomagot, és mindenki LinuxJedivé válhat (természetesen csak azok, akik leküzdik az olyan megpróbáltatásokat, mint a csomagfüggőségek erdeje).

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

Helyreigazítás

Az előző számunkban megjelent Egy kis térinformatika című cikk szerzője a tartalomjegyzék állításától eltérően Tóth Béla volt. Az érintettektől és olvasóinktól elnézést kérünk.

Programvadászat

Ehavi CD-mellékletünkön szereplő anyagaink sokkal nagyobb hangsúlyt kapnak az eddigiekhez képest, ezt a Magazin könyvtárban lévő anyagok száma és terjedelme is jelzi. A magazinban megjelenő cikkek támogatása ezután sokkal alaposabb lesz, igyekszünk minden programot (természetesen azokat, amelyek GPL felhasználási szerződés alá esnek) közzétenni, a kódok nagy részét is a megfelelő könyvtárakban találhatják meg olvasóink. Minden cikk elején a cím mellett egy kis CD-jellel jelezzük, tartozik-e CD-s anyag a témához. Olvasóink javaslatait nagy érdeklődéssel várom, és amennyiben lehetséges, eleget is teszünk az Önök kéréseinek. A javaslatokat, illetve kéréseket a cd@linuxvilag.hu levélcímre küldhetik. A következő feltételeknek kell mind a programnak, mind a levélnek megfelelnie :

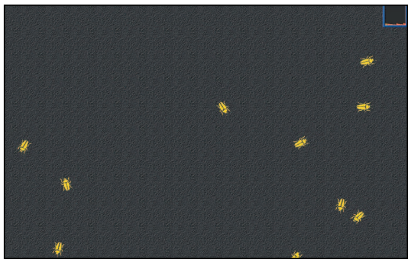
- Fel kell tüntetni a program nevét.
- A programot GPL felhasználói szerződés alatt lehessen terjeszteni.
- A program elérhetősége (URL). Ennek a közvetlen letöltési hivatkozásnak, nem pedig a program honlapcímének kell lennie.
- A program rövid leírása (minimum négy mondat, magyarul).

Kérünk mindenkit, az itt felsorolt pontok betartásával segítsen minket a munkánkban!
A megszokottaktól eltérően nemcsak a nagyobb lélegzetű anyagokat adjuk közre, hanem nagyon sok kis apróságot is, amelyekről a legtöbben nem is gondolják, hogy léteznek.

A mütyürkék

Szándékosan kezdem ezek bemutatásával, mivel a vidámság sohasem árt. Ezek a programok általában semmire sem jók, gyakorlati hasznuk nincs, hacsak azt nem soroljuk ide, hogy a felhasználót örömmel tölti el, mint egy kedves kis háziállat léte. Biztosan mindenki ismeri az Xeyes programocskát, ez egy olyan szempár, ami követi az egerünk mozgását. Bizony, bizony, arra is rá lehet venni, hogy nagyot kancsalítson! Minden Linux-kiadásnak része, az `Xfree86-tools` csomagban található.

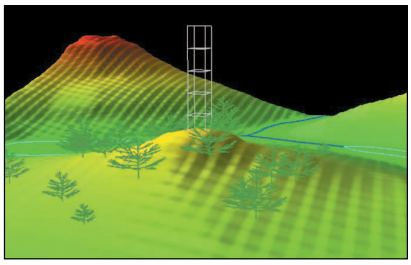
Másik nagy kedvencem az `xroach` azaz xcsótány: ha valaki nincs megelégedve a bérházakban található csótánymennyiséggel, ezzel a programmal bizony pótolhatja a hiányt.
Ha `-rc` (roach color) kapcsolóval



a bogarak színét, a `-speed` segítségével pedig a sebességüket szabhatjuk meg, a `-squish` szerintem a legéletszerűbb, az egérgombbal elkaphatjuk és halottá nyilváníthatjuk őket!

Grass5

A Grass5 (Geographic Resources Analysis Support System) segítségével háromdimenziós térképeket készíthetünk, animálhatunk, és az eredményt filmként rögzíthetjük.
➲ http://grass.ibiblio.org/



Netscape 7.0

A Mozilla-alapokon nyugvó Netscape új családjának legújabb tagja a 7-es változat, amelynek használhatóságáról megoszlanak a vélemények, de egy biztos: teljes értékű böngészőprogarmot kapunk kézhez, szinte minden bővítménnyel megtűzdelve.
➲ http://www.netscape.com

Rendszermag

A legfrissebb fejlesztői rendszermagot is közreadjuk, de használata csak a felkészült felhasználók számára ajánlott!
➲ http://www.kernel.org

Grafika

Ezután ez a könyvtár is új és állandó bérlő lesz korongunkon, mivel a Linux programjai is elérték azt a kort, hogy termelésre alkalmas munkára fogjuk őket.

Jahshaka

A Jahshaka egy mozgókép-feldolgozó és azt különféle hatásokkal ellátó programcsomag, futtatásához szükséges a Qt telepítése és OpenGL-támogatás. A jelenlegi változat Linux-, Irix- és Windows-gépeken fut és a Mac OS X-változat is hamarosan várható.
A ➲ http://www.jahshaka.com oldalon szereplő tájékoztatás szerint a fejlesztők nVidia kártyát használnak.

OpenOffice.org 1.0.1

Az irodából természetesen most sem maradhat ki az OpenOffice.org legfrissebbje, mivel igen fontosnak tartjuk az irodai alkalmazásokat, így windowsos telepítőkészlete is felkerült a korongra. A honosítókészlet ebben a könyvtárban szintén megtalálható.
➲ http://www.openoffice.org

Windows

Ebben a könyvtárban a Linux alatt használt és bevált programok windowsos változatát is közzétesszük, ezzel is segítve az átállás kínjait, mivel ha egy bizonyos programot már biztosan használunk, könnyebben szokjuk meg az új környezetet.

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

Olcsó, nagy tudású Nokia-telefonok

A világ vezető mobiltelefon-gyártójának új telefonjait hamarosan kézbe vehetjük – ha a hazai szolgáltatók is úgy akarják. Érdemes is várni rájuk, hiszen a fiatalóknak szánt, kedvező árú készülékek érdekes újdonságokkal kecsegtetnek. A 3510i és 3650 jelzésű készülékek színes kijelzővel rendelkeznek, az utóbbit beépített digitális kamerával is felszerelték. A mobiltelefon-eladások hosszú ideig töretlenül növekedtek, ám egy ideje a folyamat lelassult – a fogyasztók egyszerűen nem látják értelmét a cserének. A színes kijelző ugyanakkor a legkedveltebb fejlesztés, amit a vásárlók igényelnek, és ezért valóban hajlamosak új készülékre költeni – alighanem ezt ismerte fel a Nokia is. Mindkét új telefon alkalmas lesz MMS-ek küldésére, illetve GPRS-kapcsolatot is tudunk majd velük létesíteni. A 3510i ára körülbelül 250 euró lesz, a 3650-esért pedig nagyjából 500 eurónak megfelelő összeget kell majd leszurkolni, ha előfizetést nem vásárlunk mellé.

➲ http://www.nokia.com

Távoli felügyelet egyetlen lapkával

A QLogic – a cég elsősorban tárolóhálózatokhoz fejleszt eszközöket – olyan Remote Management Controller (RMC) lapkákat jelentett be, amelyek tudása a felügyeleti kártyákéval vetekszik. A cél a toronyba szerelhető és kisméretű kiszolgálók piacának meghódítása, hiszen ezeknél PCI-foglalat híján nincs lehetőség külön felügyeleti kártya beszerelésére a gépbe. A távoli felügyelet, az újraindítás, a programtelepítés és frissítés, a hibajavítás lehetősége nem új, ám eddig leginkább a felső kategóriás kiszolgálóknál alkalmazták. Az új, az Intelligent Platform Management Interface (IPMI) előírások 1.5-ös változatát ismerő lapkákkal viszont olcsón elérhetővé válik a távoli felügyelet lehetősége, amelyet ezentúl remélhetőleg a belépő szintű gépeket vásárlóknak sem kell nélkülözniük.

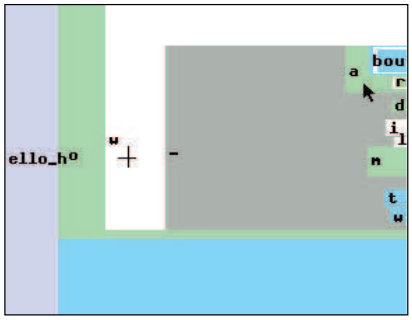
➲ http://www.qlogic.comdeveloper.intel.com/design/servers/ipmi/

Feltámadóban a Midnight Commander

Az utolsó, 4.5.55 számot viselő kiadás majdnem egy éve látott napvilágot, azóta tetszhalott állapotba került a Midnight Commander. Nemrég azonban megjelent a 4.6-os változat előzetes kiadása, amely elsősorban a fennálló hibák javítását célozza – újabb szolgáltatások beépítésére csak ezután lehet gondolni. A cél tehát a különféle biztonsági és üzembiztossági hiányosságok javítgatása volt, illetve a leírást két új nyelven, spanyolul és oroszul is el lehet érni.

Szemmel gépelés esete forog fenn

A betegséggel küzdő vagy mozgássérült emberek számára sokszor súlyos nehézséget jelent a billentyűzet és az egér használata, eddig mégsem sikerült igazán jó helyettesítő eszközt találni.
A Dasher tervezet keretében olyan alkalmazás fejlesztését tűzték ki célul, ami a felhasználó szemmozgását követve teszi lehetővé szövegek bevitelét, illetve az



egérmutató mozgatását. A program egy – bármilyen, erre alkalmas – kamerával követi a monitor előtt ülő szemének mozgását. A felhasználó a megfelelő pontra bámulva a monitoron kijelölheti a kívánt betűt, majd a képernyő másik oldalára nézve hozzáadhatja a már „beírt" karakterekhez. A program a képződő szót elemezve próbálja meg kitalálni, hogy a felhasználó mit szeretne beírni, és kiemelve felkínálja a várható választásokat. Használatának elsajátítása természetesen némi gyakorlást igényel, ám ezt követően a segítségével percenként 25 szó is beírható. További előnye, hogy az emberek számára – a gépeléssel ellentétben – természetes viselkedés, hogy oda néznek, ahova nyúlnak vagy amerre haladnak. A Dasher az egészséges emberek számára is jól jöhet, ha mobiltelefonba vagy zsebtitkárba akarnak szöveget bevinni, és gyakorlatilag bármely nyelvet képes támogatni.
A tervezetben résztvevők munkájának eredménye nyílt forrással érhető el.
➲ http://www.inference.phy.cam.ac.uk/dasher/

Levélszemét-szégyentábla

„Az oldalnak az az alapvető célja, hogy az elrettentés eszközével próbálja visszaszorítani a kéretlen leveleket küldők hadát." „Legyen ez az oldal egyfajta mementója annak, hogy egyes szolgáltatók és szolgáltatások fennállásuk történetében legalább egyszer figyelmen kívül hagyták azokat az alapvető bizalmi elveket, amelyekre pedig szolgáltatásuk, termékeik kínálásakor oly gyakran és szívesen hivatkoznak."
Kemény szavakkal fogalmaz a magyar levélszemét-szégyentábla írói képességekkel is megáldott munkatársa. Csak remélni tudjuk, hogy oldaluk mind nagyobb nyilvánosságot kap, így valóban szégyen lesz felkerülni rá – legyünk számítók –, és divatos lesz nem rajta lenni. Az egyelőre meglehetősen puritán kinézetű, lassúcskán betöltődő oldal akár komoly adatbázissá is fejlődhet, amihez csak kitartást és lankadatlan munkakedvet kívánhatunk fenntartóinak.
➲ http://antispam.freeweb.hu

Öreg böngésző nem vén böngésző

Ha lenne öregek klubja programok számára, a Netscape 4.x sorozat minden bizonnyal tiszteletbeli tag lenne. Készítői nyilván nem kedvtelésből frissítgetik a csomagot, így alighanem még mindig van igény az öregecske böngészőre. A sorozat legújabb tagja a 4.8-as változat – amelyben semmi érdekesebb újdonságot nem sikerült találni –, ami elődjeihez hasonlóan MacOS, Linux, illetve Unix és Windows operációs rendszerek alá érhető el.
➲ http://www.netscape.com

750 MB-os zipmeghajtó

Az Iomega új, 750 MB kapacitású meghajtójáról nem sokat árulnak el a gyártó honlapján – leginkább csak azt, hogy létezik. A külső meghajtó USB 2.0-s kapun keresztül csatlakozik a számítógéphez, sebessége 7 MB másodpercenként. Kapunk hozzá egy önműködő mentést végző programot is, így ütemezett biztonsági mentésekhez is célszerűen használható. A meghajtó az ősz folyamán FireWire felületű változatban is megjelenik. Képes a korábbi zipmeghajtók által használt 100 és 250 MB-os lemezek kezelésére is, ám az előbbieket csak olvasni tudja. Az ára körülbelül 50 000 forint lesz, ha Magyarországra is megérkeznek az első példányok.
➲ http://www.iomega-europe.com

Asus kézigép

A hordozható számítógépek piacán már jól hangzik az Asus név, ám a cég tovább terjeszkedik. Nemrég bemutatta első tenyérgépét, amely a MyPal A600 Pocket PC nevet nyerte a keresztségben. Az Asus a világ legkönnyebb, legvékonyabb, ennek ellenére mégis legnagyobb teljesítményű kézigépének nevezi, ami 75×125×12,8 mm-es méreteit, 138 grammos súlyát és 400 MHz-es Intel-processzorát figyelembe véve nem is hangzik olyan hihetetlenül. A gépecske 64 MB SDRAM és 32 MB flashmemóriával gazdálkodik, és ha ez nem volna elég, CF-foglalatát használva további memóriakártyákkal vagy akár IBM Microdrive-meghajtóval is bővíthetjük. Megfelelő kiegészítővel internetezésre, műholdas helymeghatározásra is használhatjuk.

➲ http://www.asus.com.tw

Megjelent a Jungo OpenRG v2.0

A Jungo Software Technologies bejelentette OpenRG nevű beágyazott Linux alaprendszerének 2.0-s változatát. A rendszert a cég elsősorban otthoni hálózati átjárók fejlesztői számára készíti. Az új változat Universal Plug and Play modullal is rendelkezik, így az őt futtató készülékek telepítése – öntanító képességének és kiterjedt protokolltámogatásának is köszönhetően – valóban egyetlen mozdulattal intézhető el. A rendszer grafikus felületen keresztül állítható be, és a készülék a hálózat térképének megjelenítésére is képes. További érdekessége az egyszálas működés. A hagyományosnak tekinthető, több szálat futtató készülékeknél minden egyes hálózati kapcsolat kezeléséhez új szál indul, ami saját, hozzávetőlegesen 150 KB méretű memóriaterületet foglal. Az egyszálas működésnél ellenben egyetlen programszál szolgál ki minden kapcsolatot, így kisebb a processzor terhelése, és az újabb kapcsolatok kiszolgálásához is csak 760 bájt memóriára van szükség. Meglepő, hogy a Jungo látszólag az árral szemben úszva rúgja fel a többszálas rendszerek hagyományát, ám az, hogy így olcsóbb, kisebb teljesítményű processzor és kevesebb memória is elég a készülékek működtetéséhez, meggyőző érvnek tűnik.

➲ http://www.jungo.com

Webre minden mosógépet!

Az IBM és az USA Technologies összefogásának köszönhetően az amerikai egyetemeken és főiskolákon a mosógépek hamarosan hálózati kapcsolatot is tudnak majd létesíteni. A webes hűtőgép ötlete már régen nem újdonság, ám a két cég talán mégsem csupán feltűnési viszketegségtől vezérelve dobta be a dolgot. A fejlesztések révén először is megszűnik az odaát megszokott pénzbedobálós fizetési rendszer, az esedékes díjat egy azonosító kártyával vagy mobiltelefonon keresztül rendezhetik a diákok. Emellett a gépek közös hálózathoz csatlakoznak, az érdeklődők pedig Weben keresztül ellenőrizhetik, hogy mikor válik szabaddá valamelyik mosógép. Különféle mosási adalékok közül is válogathatnak, a mosási művelet befejezéséről pedig személyhívón vagy elektronikus levélben kérhetnek értesítést. A hálózat lehetővé teszi a gépek állapotának, működésének megfigyelését is, a folyamatos felügyelettel számos meghibásodást meg lehet majd előzni. A tengerentúlon egyre inkább terjednek a vezeték – és hitelkártya – nélküli fizetést lehetővé tévő megoldások, a különféle apró vásárlások összege az előjelzések szerint 2005-re 200 milliárd dollárt tesz majd ki. Mivel a gépek egyre kevésbé tárolnak aprópénzt, a feltörésükre tett kísérletek által okozott károk is jelentősen csökkenthetők.

Elkészült a United Linux első próbaváltozata

A The SCO Group, a Conectiva S.A., a SuSe Linux AG és a Turbolinux közös munkájának eredményeként megszü-

UNITEDLINUX

letett a United Linux első próbaváltozata. A terjesztés egyelőre csak zárt körben érhető el, nyilvános kiadás valamikor szeptember végén várható. A United Linux életre hívásának célja, hogy egységes terjesztést készítsenek, amelyre építve – elsősorban az üzleti világ igényeit szem előtt tartva – könnyebb a linuxos alkalmazások fejlesztése, a különféle tanúsítványok kibocsátása, illetve a világméretű értékesítés és támogatás megvalósítása. A terjesztés amellett, hogy önmaga is egyfajta szabvány lenne, az LSB- és Li18nux-előírásoknak is megfelel. A legnagyobb világnyelvek mellett magyarul is elérhető lesz.

➲ http://www.unitedlinux.com

Oracle fürtözött Linux-fájlrendszer ingyenesen

Az Oracle bejelentette, hogy forráskódként, GPL szerződéssel is közzéteszi fürtözött Linux-fájlrendszer megoldását. A forráskód máris letölthető az Oracle internetes fejlesztői hálózata, az Oracle Technology Network (OTN) webhelyéről. Az új fürtözött Linux-fájlrendszer fejlesztői változata a cég várakozásai szerint leegyszerűsíti az Oracle9i Real Application Clusters rendszerek felhasználói számára a fürtözött adatbázis-rendszerek kezelését és felügyeletét.
➲ http://www.oracle.com

Dobd el, lebomlik!

A Sony WM-FX202 jelzésű sétálómagnója november elsejétől szokatlanul környezetbarát kivitelben lesz kapható – készülékháza ugyanis túlnyomórészt biológiai úton lebomló műanyagból készül. A különleges műanyag – a hagyományos műanyagoktól eltérően – növények feldolgozásával készül, a korábbiak előállításánál kőolajat használtak fel. Amikor a készülék a szemétbe kerül, házának anyaga széndioxidra és vízre bomlik le, egyéb kellemetlen tulajdonságai viszont nincsenek – érdekessége éppen az, hogy környezetbarát jellege ellenére tartós, hő- és ütésálló és könnyen formázható. Arról nem szól a hír, hogy a készülék egyéb alkatrészei mennyire kímélik a természetet, illetve hogy a különleges műanyag gyártásakor vajon mennyi szennyeződés keletkezik.

Ugyancsak a Sony érdekessége az a videomagnó, amely a tulajdonosának ízlése szerinti műsorokat vesz fel. A készülék folyamatos internetkapcsolatot kíván, a műsort a hálózatról tölti le, és a megadott kulcsszavak alapján válogatja ki, hogy mely műsorokat érdemes rögzítenie. Némi használat után arra is képes, hogy igazodjon felhasználójának szokásaihoz, ízléséhez. A készülék nem szalagokkal, hanem egy 160 GB méretű merevlemezzel dolgozik, amelyre minőségtől függően 15–100 órányi anyagot tud rögzíteni. A Sony olyan szolgáltatás indítását is tervezi, amelynek segítségével a vásárlók távolról, mobiltelefonról is beprogramozhatják az eszközt. Az új videó egyelőre csak Japánban lesz kapható, ami negyedmillió forint feletti árát tekintve talán nem is nagy meglepetés.

Internet SCSI-rendszerek az Adaptec kínálatában

Az Internet terjedésével az IP-protokollt is egyre több területen használják, és nem maradhatnak ki a sorból a tárolóhálózatok (SAN) sem. Az Adaptec – más neves gyártókkal együttműködve – iSCSI, azaz internetalapú SCSI-csatolókkal bővíti kínálatát. Az új csatolók feladata az, hogy a SCSI-sínen utazó csomagokat szabványos ethernet hálózati csomagokká alakítsák, amelyek ezt követően bármely ethernethálózaton át, tehát a meglévő hálózati elemek felhasználásával is továbbíthatók. Az iSCSI-alapú adattárolás – az olcsó ethernetösszetevőknek köszönhetően – kisebb teljesítményű alkalmazásoknál olcsóbb a hagyományos FC-megoldásoknál. További előnye, hogy a tárolóeszközök teljesítőképessége leállás nélkül bővíthető, a végpontok között több útvonalat kiépítve magas fokon hibatűrő, illetve az IP-protokoll elterjedtsége révén különböző operációs rendszereket futtató, akár eltérő típusú gépekről is közösen lehet használni ugyanazt a tárhelyet.
➲ http://www.adaptec.com

Alakulgat az UHU-Linux

Az első magyar terjesztés ugyan még nem érett be, de az egyik oldala már kezd pirosodni. A fejlesztők túlléptek a próbaváltozatok korszakán, és augusztus közepén megjelent az RC-1 kiadás. A terjesztés szabadon letölthető az UHU-Linux FTP-kiszolgálójáról, valamint a 14. oldalon beszélgetét olvashatnak a készítőkkel, illetve meg is rendelhető – az utóbbi esetben csak a postaköltséget kell állni.
➲ http://www.uhulinux.hu

Bővít a FreeStart

A FreeStart 2002. augusztus 7-én megkezdte szolgáltatási területének jelentős bővítését. Az eddigi három mellett az év végére befejeződő fejlesztések eredményeképpen újabb 19 körzetben lesz elérhető a cég ingyenes internetszolgáltatása. A bejegyzett felhasználók nemcsak hálózati hozzáférést, de levélcímet és webtárhelyet is kapnak a FreeStarttól. Fontos változás, hogy megszűnt a kapcsolatok óránkénti megszakítása is, amelyet még februárban kényszerültek bevezetni.
➲ http://www.freestart.hu

Samsung telefon Palm OS rendszerrel

A Samsung új mobiltelefonja egyszerre telefon és zsebtitkár. Fontos előnye, hogy a zsebtitkároknál megszokott méretű kijelzőjére Palm OS rajzolgatja az elemeket, így a kezelése nem okozhat gondot senkinek, aki találkozott már a világ legnépszerűbb tenyérgépen futó operációs rendszerével. Az SPH-I300 természetesen képes a Palm OS-alapú alkalmazások futtatására, kijelzője 256 szín megjelenítésére alkalmas, valamint internetezni és levelezni is lehet vele. Megtalálhatók benne a megszokott Palm-alkalmazások, mint címlista, határidőnapló, tennivalók, jegyzettömb és számológép. Az adatok és programok tárolására 8 MB memória áll tulajdonosa rendelkezésére, mérete 125×58×21 mm, súlya pedig 170 gramm.
➲ http://www.samsungelectronics.com

Svédország is a szabad programok iránt érdeklődik

Hosszú már a sora azon országoknak, ahol az államigazgatásban komolyan foglalkoznak a szabad programok használatával. Svédország is csatlakozik hozzájuk, annak ellenére, hogy nemrég írt alá egy szerződést a Microsofttal. A fő ösztönző erő természetesen a takarékosság, ami az említett szerződésben szereplő félmilliárd koronás összeget figyelembe véve nem meglepő. A frissen létrehozott, a Linux használatából várható előnyök értékelését végző munkacsoport tevékenysége még csak nemrég kezdődött, ám azt már most tudni lehet, hogy a helyi rendőrség, a munkaügyi hivatal és az adóhivatal is résztvesz benne. A Linux egyre nagyobb szerephez jut Dánia és Norvégia hivatalaiban is, de leglátványosabb bázisa a távol-keleti országokban van: Koreában százezer feletti a linuxos asztali gépek száma, Kína pedig egyenesen saját terjesztést fejleszt.

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

Cégvilág

SMS-ben is tájékoztatja utasait a KLM

A KLM Holland Királyi Légitársaság ingyenes SMS-szolgáltatást vezetett be törzsutasai számára. Az i-Cell

Kft. által fejlesztett és üzemeltetett i-Flight rendszer segítségével a KLM magasabb szintű kiszolgálást, kényelmesebb utazást teremt utasainak. A szolgáltatás igénybevételével a Flying Dutchman-tagok tájékoztató SMS-üzenetet kapnak mobiltelefonjukra a Budapestről induló, illetve Budapestre érkező KLM-járataik helyzetéről: várható indulási, illetve érkezési idejükről, valamint az esetleges járattörlésekről. Az értesítéseket magyar, angol, német, francia és holland nyelven lehet kérni, a szolgáltatást a KLM weboldalán vagy jegyirodáiban kitöltött formanyomtatvány kitöltésével lehet igényelni, kizárólag magyarországi mobilszolgáltatóhoz tartozó telefonszámra.
➲ http://www.klm.hu

Távképzés a Vám- és Pénzügyőrségnél

A veszprémi székhelyű Netura Kft. által fejlesztett és üzemeltetett internetes EU-tanfolyamot választotta a Vám- és Pénzügyőrség hétezer fős személyi állományának továbbképzésére. A tantermes oktatással szemben a választáshoz nagyban hozzájárult, hogy az online tanfolyam könnyen, az ország bármely

területéről elérhető, tananyaga letölthető, és a vizsgák tetszés szerinti időpontokban letehetők. A tanfolyam tematikáját és tananyagát a Netura Kft. állította össze az Európai Bizottság Magyarországi Delegációjának szakmai támogatásával. A tanfolyam során a hallgatók általános betekintést nyernek az Európai Unió történelmi hátterébe, időrendjébe, intézményrendszerébe, külkapcsolati és jogi rendszerébe, valamint számos más vonatkozású témába. A hallgatók a rendszeres online vizsgakérdések megválaszolása mellett záróvizsgát is tesznek. Egy az Európai Unióval kapcsolatos, támogatott, általános tanfolyam bárki számára ingyenesen elérhető az Interneten, a
➲ http://www.eutanfolyam.hu címen.

Új tulajdonoshoz került a DataNet

Az Antel Holdings tulajdonába került a KPNQwest Ebone Central Europe, illetve vele együtt hazánk egyik legrégebben működő internetszolgáltatója, a korábban a GTS által felvásárolt DataNet, mai nevén GTS-DataNet. A KPNQwest a legnagyobb választható távközlési szolgáltató Közép-Európában, kínálatában hang- és adatátviteli, IP-alapú, valamint különféle széles sávú hálózati szolgáltatások találhatók, ügyfélköre a magánszemélyektől a nagyvállalatokig terjed.
➲ http://www.datanet.hu

IP-telefónia a DÉMÁSZ-nál

A DÉMÁSZ Rt. nemrég bővítette gerinchálózatát, ám elavult telefonrendszerének korszerűsítése még váratott magára. Az áramszolgáltató megbízását a Synergon Rt. nyerte el, amely egységes hang- és adattovábbító hálózatot épít ki, azaz a továbbiakban a szolgáltató belső telefonforgalma is IP-alapon folyik majd. A rendszer 2003 nyarára várható teljes kiépítésével nagyobb IP-alapú telefonhálózat jön létre, mint a Magyarországon eddig telepített hasonló hálózatok összessége.
➲ http://www.synergon.hu

SMS-küldés vezetékes telefonról

Szeptember 1-jével vezette be a Matáv a vezetékes SMS-küldés és -fogadás lehetőségét. A szolgáltatás egyelőre csak Matáv-előfizetők között, illetve Matáv-Westel és visszirányú viszonylatban működik, itt viszont telefonra, faxkészülékre, hangpostára egyaránt küldhető üzenet – feltéve, hogy megfelelő készülékkel rendelkezünk. Ilyet kétféle típusból választhatunk a cég kínálatából, 25 és 30 ezer forintos áron. Ha nem akarunk ennyit szánni új készülékre, fogadni még mindig tudjuk az üzeneteket, ilyenkor a rendszer felolvassa nekünk a szöveget.
➲ http://www.matav.hu

Kitüntetés Björn Johan Flakstadnak

Görgey Gábor kulturális miniszter a Pro Cultura Hungarica kitüntetést adományozta *Björn Johan Flakstad*-nak, a Pannon GSM leköszönő vezérigazgatójának. Flakstad folyamatosan támogatta a magyar képzőművészetet, sokat tett annak nemzetközi megismertetéséért. A szolgáltató 1995 óta többek közt kiállítások szervezésével, képek vásárlásával, helyreállításával és kiadványok támogatásával is segítette a Magyar Nemzeti Galériát.
➲ http://www.nkom.hu

Sikeres az ArchiCAD az amerikai parti őrségnél

A Graphisoft bejelentette, hogy az amerikai parti őrség ezentúl a vállalat termékét, az ArchiCAD-et alkalmazza stratégiai létesítményeinek hatékonyabb kihasználására. A Graphisoft nemcsak építési és kivitelezési tervezésre, hanem a meglévő épületekről készített háromdimenziós modellek felhasználásával azok jobb helykihasználásának kidolgozására is használható. Használatával többek között könnyebben lehet határozni a legfontosabb beruházásokról, csökkenthetők a fenntartási költségek és felderíthetők a sebezhető pontok.
➲ http://www.graphisoft.hu

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

Linux a tanteremben – 3. rész

Amennyiben a jövő szempontjából értékesnek tartjuk a Linuxot, a jövő nemzedékét kell felkarolnunk, hogy elsajátítsák a Unix-látásmódot, és megtanulják kezelni a Unix-rendszereket.

A sorozat zárásaként a szegedi Radnóti Miklós Gimnáziumban az elmúlt tanévekben tartott Linux-szakkörökről szóljon egy rövid beszámoló.
„Nálunk több szakkörpróbálkozás is volt, és mindegyik dugájába dőlt" – olvastam nemrég a SuLinux levelezési listán egy neves linuxos rendszergazda-tanár sorait. Nálunk a tanév eleji lelkesedés sokszor nagyobb volt, mint a későbbi részvétel: a gyerekek között ugyanúgy „divat" a Linux használata, ahogyan az különc módon öltözködni; azok, akik azonban komolyan is gondolják, hogy hátat fordítanak a Microsoft-világnak, valójában kevesen vannak. Szerencsére már nem mindenütt, az idézett rendszergazda-tanár középiskolájában a nyári szünet elején tíz nap alatt száznál is több különböző felhasználó jelentkezett be a Linux-kiszolgálóra. Nálunk ez a szám a teljes nyári szünet alatt öt alatt volt…
A Linux-szakkört azért többen látogatták, csakhogy szinte mindig mások. A gyerekek tudásszintje egymáshoz képest lényegesen különböző volt, így lehetetlen volt olyan órát tartani, amiből mindegyikük nyert volna valamit. Többször is telepítettük a Linuxot (először még abban az időben, amikor a 6.1-es SuSE kijött), tanítottam könnyebb héjprogramozást a *dialog* program bemutatásával, egyszerűbb programokat készítettünk együtt Kylixban és PHP-ben (PostgreSQL-támogatással), (a legtöbb anyagot az ➲ http://ftp://pc10.radnoti-szeged.sulinet.hu/home/kovacsz/Linux címen nyilvánossá tettem). A gyerekek számára mindez nem volt kivételesen nagy élmény, azonban remélem, néhányójuknak induló lökés is lehetett. A linuxos tudást a gyerekek azonban nem a szakkörön gyűjtik össze, hanem saját tapasztalataik alapján az otthoni számítógépük előtt.

A projektmunka és a Linux

Ezidáig a tehetséges gyerekek tanításának legösztönzőbb eszközét a közös játékprogram-fejlesztésben találtam meg. Még középiskolás voltam, amikor számítástechnika tanárom ugyanezzel a módszerrel indított el bennünket a véresen komolyan vett programfejlesztés felé, abból a célból, hogy hatékonyan készüljünk fel az Országos Középiskolai Tanulmányi Versenyre. Az egyetemen is volt olyan gyakorlatvezetőm, aki programozás tárgyból kötelezően beadandó programként logikai játék

```
#include "allegro.h"

BITMAP *kocsi, *hatter;
 /* Az aut  Øs a hÆttØr. */
PALLETE paletta; /* Az Øppen ØrvØnyes
➥sz npaletta. 255 sz nß, a 0. Ætlætsz  */


void mozgat()
 {

 BITMAP *megjelenitendo; /* Ebben tÆroljuk
➥a kØpernyı vÆltoztatand  rØszØt. */

 char c; /* A megnyomott gomb. */
 int lepeskoz=1; /* Az aut  mozgÆsÆnak
       ➥gyorsasÆga. GØptıl is f gg. */

 int kepatlo,irany=192;
 /* Aut  "ÆtmØrıje", indul  irÆny (Øszak). */
 float x,y,iranyx=-1,iranyy=0;
 /* Aut  koordinÆtÆi, indul  irÆnya. */
 kepatlo=sqrt((kocsi->w)*(kocsi->w)+
           ➥(kocsi->h)*(kocsi->h));
 // Pitagorasz-tØtel!
 kepatlo+=5; /* Egy kicsit megn velj k, hogy
            ➥fordulÆsnÆl is szØp legyen. */

megjelenitendo=create_bitmap(kepatlo+lepeskoz*2,
➥kepatlo+lepeskoz*2);

 /* AzØrt adjuk mØg hozzÆ a lØpØsk zt, hogy
    ➥nagy gyorsasÆgnÆl ne fussunk ki. */

 x=(320-(megjelenitendo->w))/2;
 /* BeÆll tjuk az aut  indulÆsi poz ci jÆt. */
 y=(200-(megjelenitendo->h))-40;

 do
 {

  if (key[KEY_LEFT]) irany-=2; /* balra */
  if (key[KEY_RIGHT]) irany+=2; /* jobbra */
  if (key[KEY_UP])
 {
 /* elıre */
   y-=((float)lepeskoz*iranyy);
   x-=((float)lepeskoz*iranyx);

   }

  if (key[KEY_DOWN])
 {
 /* hÆtra */
   y+=((float)lepeskoz*iranyy);
   x+=((float)lepeskoz*iranyx);

   }

iranyx=cos((float)((((float)irany+64)/128)*3.1415));
 /* koszinusz... :-) */
  iranyy=sin((float)((((float)irany+64)/128)*
➥3.1415));
 /* szinusz... :-) */
  /* Feltessz k a hÆttØr megfelelı darabjÆt
  ➥a vÆltoztatand  rØszre: */

blit(hatter,megjelenitendo,(int)x,(int)y,0,0,
➥kepatlo+lepeskoz*2,kepatlo+lepeskoz*2);
```

A lista folytatása a következő oldalon található

elkészítését írta elő. Érdekes, hogy ez az ötlet sokszor már a 13 éves gyerekeknél is bejön, habár logikai játék helyett eddig mindhárom alkalommal egy-egy akciójátékot kezdtünk el írni.
Az első próbálkozáson öt évvel ezelőttre tehető. Ekkor még csak próbaképpen tartottam szakkört a Radnótiban, negyedéves egyetemistaként. A szakkört a tanév közepén indítottam, és megbeszéltük a négy, nálam néhány évvel fiatalabb fiúval, hogy valami igen komoly stratégiai játékot írunk Turbo Pascalban, assembly-betétekkel. Néhány segédeljárás el is készült, összeállt némi grafikai terv is, de a program sohasem került futtatható állapotba.
A második próba három évvel ezelőtt volt, amikor a „A vágtázó halottkémek" fedőnevű projektben hét tanulóval egy ténylegesen működő, animált mozgást is tartalmazó, DOS és Linux alatt is futtatható kódot dobtunk össze, sok grafikával. A játék „Radnótis kaland" néven futott, és a lényege pedig az volt, hogy a főhőssel a suliban mászkálva kell mindenféle feladatot teljesíteni. Két szakkörös gyereket fényképeztünk le több fázisban a fehér fal előtt, majd a fényképeket beolvastuk, a Gimppel „megfésültük", s ezután a megfelelő C nyelvi utasítással a gimnázium szintén beolvasott jellemző részletei elé másoltuk őket (➲ //ftp://pc10.radnoti-szeged.hu/home/kovacsz/C_szakkor/JÁTÉK!).
Ez a második projekt tehát már C nyelven futott. A DOS-os DJGPP (➲ http://www.delorie.com/djgpp) alatt az Allegro-csomag (➲ http://allegro.cc) segítségével írtuk a programkódot, a szövegkiíró és a perspektív megjelenítésért felelős eljárást pedig egy-egy 14 éves tanítványom alkotta. Habár a programnak csak egy „nagyon példa" változata lett készen, a projektet mindenképp sikerként könyvelem el. Az Allegro csomagot sajnos kevesen ismerik; ez egy könnyen használható, gyors programfejlesztést lehetővé tévő eljárásgyűjtemény, amelynek fő erőssége a gyors grafikai megjelenítés és a több felületen való felhasználhatóság.
Annyira gyorsan lehet benne fejleszteni, hogy tavaly egy olyan projektet indítottam benne el, ami csak hetvensoros volt, mégis már így is működő autóversenyprogram lett. Ma is szívesen játszanak vele kicsik és nagyok egyaránt.
Ezt a programot kezdtük aztán Maxirace néven fejleszteni (a nevet egy hetedikes tanítványom adta) a SourceForge-on, CVS-támogatással. Ez utóbbihoz már Linux kellett; bár DOS alatt is lehetett volna CVS-t

A lista folytatása az előző oldalról

```
  /* A vÆltoztatand  rØszre rÆrajzoljuk az aut t: */

rotate_sprite(megjelenitendo,kocsi,lepeskoz+
➥(kepatlo-kocsi->w)/2, lepeskoz+(kepatlo-
➥kocsi->h)/2,itofix(irany));

  sync(); /* Ez a sor biztos tja az animÆci
         ➥villogÆsmentessØgØt. */

draw_sprite(screen,megjelenitendo,(int)x,(int)y);
 /* KIRAJZOL`S! */
  rest(5); /* 5 ezredmÆsodperc vÆrakozÆs. */
  }
 while (0!key[KEY_ESC]);
 /* KilØp nk, ha ESC volt a megnyomott gomb. */
 }


void inicializalas()
 {

 allegro_init();
 /* Ez minden grafikus program elejØre kell. */
 install_keyboard();
 /* Ez mindig kell, ha a billentyßzetet
    hasznÆljuk. */
 install_timer(); /* Ez mindig kell, ha
                     idız tØst hasznÆlunk. */
 hatter=load_bitmap("hatter.bmp",paletta);
 /* HÆttØrkØp beolvasÆsa. */

 kocsi = load_bitmap("auto.bmp", paletta);
  /* Kocsi beolvasÆsa. */

 set_gfx_mode( /* Grafikus m d beÆll tÆsa. */
#ifdef LINUX
  GFX_XWINDOWS,  /* Linuxon X Window alatt
                ➥fusson a program, */
#else
  GFX_AUTODETECT, /* egyØbkØnt legyen az
                ➥alapØrtelmezett m d. */
#endif
  320, 200, 0, 0);
 /* FelbontÆs. */
 set_pallete(paletta);
 /* Sz npaletta beÆll tÆsa. */
 draw_sprite(screen,hatter,0,0);
 /* Kirajzoljuk a hÆtteret. */
 }


main()
 {
 /* Fıprogram. */
 inicializalas();  /* Megh vjuk az
 ➥inicializalas() rØszprogramot. */
 mozgat();
 /* Elkezdj k a tØnyleges jÆtØkot. */
 }

#ifdef LINUX
END_OF_MAIN(); /* Ez Linuxon kell,
               ➥DOS alatt nem. */
#endif
```

telepíteni, a 7.3-as SuSE-ban minden konyhakészen rendelkezésre állt, az Allegróval együtt. A Maxirace program fejlesztésébe ezután sok apró fiatal kapcsolódott be: néhányan weboldalt készítenek hozzá (részint Windows alatt), mások háttereket és kocsikat rajzolnak, a legügyesebbek pedig a kódba is belenyúlnak. A CVS logikáját még nem sikerült megszerettetnem: környezeti beállításaink még nem elég kényelmesek ahhoz, hogy önműködően *update*-tel és *cvs commit*-tal tartsuk a kapcsolatot a CVS-kiszolgálóval (először be kell `ssh`-zni a *pc9*-re, csak onnan lehet elérni a SourceForge kiszolgálóját, majd a letöltés után a friss anyagot át kell másolni a grafikus munkaállomásra, végül a módosított fájlokat vissza kell tenni a *pc9*-re, s csak ezután indulhat a visszatöltés a raktárba). Ellenben rengeteg apróságot és szépséget el lehet mondani a gyerekeknek, mialatt a „nagy cél" elérése érdekében kis célokat tűzünk ki, s azokat oldjuk meg. A hetente tartott kétórás szakkör magját egy ötfős tevékeny csoport alkotja, hozzájuk verődik 2–3 főnyi „kíváncsiak társasága", azonban a hivatalos fejlesztői csapat nyolctagú: az egyik csapattag 19 éves, a többiek viszont mind 13–14 évesek. Legalább tíz fő a további „pártoló tagok" száma, ezen felül az iskola tanulóinak egy része véletlenül is elindítja olykor a Windows alatti változatot.

Allegro, azaz vidáman, gyorsan

A listán olvasható a 0.2-es változat forrása, ami igazából a legelső változat, csak megfésült, megjegyzésekkel ellátott formában. A jelenlegi, a ➲ http://sf.net/projects/maxirace címről letölthető ennél már ügyesebb, és a kód is profibb formájú. Már ez a programlista is jól mutatja azonban az Allegro egyszerűségét, és a

```
gcc -DLINUX -o maxirace.exe
➥maxirace.c 'allegro-config --libs'
```

parancssorral gond nélkül lefordul. A kimenet a *maxirace.exe* lesz, a bináris kódot a gyerekek miatt a futtathatóságot hangsúlyozandó *.exe* kiterjesztéssel hozzuk létre; az utolsó órákon a kiterjesztést már elhagytam. Az `allegro-config --version` parancssorral vizsgálhatjuk meg, hogy Allegro-csomagunk kellően friss-e a fordításhoz; nekem a program a 3.9.37-es változattal gond nélkül lefordult. A pillanatnyi könyvtárban helyezzünk el egy *hatter.bmp* nevű 320×200-as képet (lehetőleg kacskaringós versenypályával) és egy ennél jóval kisebb *auto.bmp*-t, ami jól elfér a versenypályánkon. Ezután a játék már indítható is: autónkat a nyílbillentyűkkel lehet irányítani az autóverseny-programokban megszokott módon, kilépni az Esc-pel tudunk. Programunk X-ablakban fut, de a `GFX_XWINDOWS` átírásával megoldható, hogy például Svgalib alatt fusson. Ha DOS alá kívánunk fordítani (DJGPP-vel, elegendő hozzá a 2.01-es változat), módja:

```
gcc -o maxirace.exe maxirace.c
➥-lalleg
```

A Maxirace kódja kellően rövid ahhoz, hogy ne rettentse el a kezdő programozókat, de elég hosszú és elég nyitott ahhoz is, hogy egy néhány hónapos programozói alapképzéshez ugródeszka legyen. Emellett nemcsak autóverseny-program kiinduló alapja lehet: a kétdimenziós játékok egy része egy tengely körüli forgásra és előre-hátramenetre épül, az alapprogram tehát számos irányba továbbfejleszthető.

Az idén januártól már nem is Linux-, hanem Maxirace-szakkört hirdettem a gyerekeknek. Nagyon jó hangulatú műhelymunkában volt részünk ezekkel a fiatalokkal hétről hétre, és ha lassan haladtunk is előre, az eredmény magáért beszélt.

E sorok írásakor, augusztus végén azt tervezem, hogy a programot ősztől kezdve továbbfejlesztjük. A SourceForge-os projektbe természetesen bárki bekapcsolódhat, aki ezek után kedvet kapott hozzá. Tanítványainkkal talán mégis a legjobb új projektet kezdeni, akár a SourceForge-on, akár saját CVS-kiszolgálón, akár CVS-kiszolgáló nélkül. Megítélésem szerint az öröm forrása a közös munkában a létrehozott termék varázsában van, abban, hogy mindenki megtalálhatja a hozzá illő, képességeinek megfelelő részfeladatot, de ha kíváncsi mások munkájára, az sincs elrejtve előle.

Zárszó

A nyílt forráskódra épülő rendszereknek számos előnyük van, de az oktatói-nevelői munka kapcsán egyet mindenképpen ki kell emelnem: a tájékoztatás, a tudás szabadságát. A zárt forrású kereskedelmi programok meghatározásukból adódóan megfosztják a felhasználót attól, hogy megtudja, „mitől is megy" a program. Ma már olyan bonyolult rendszerekre van szükség, hogy akármilyen ügyes is egy projekt (a legjobb szakembereket gyakran a Microsoft vásárolja fel), tökéletes programot alkotni képtelenség. Ez az adat visszatartásának zsákutcája. Ha egy tanár szakmailag felkészült szeretne maradni, félmegoldás, ha csak annyit mond: „nem tudom, miért nem működik" vagy „fogalmam sincs, mitől fagy le folyton". A nyílt rendszerek tanulmányozhatók, és a tanulás legmélyebb formáját, az önképzést segítik elő. Valljuk be, diákjaink némelyik területen sokkal többet tudnak a Linuxról, mint mi, a tanáraik. Sajnos sok tanár éppen a szakmai hírnevét félti, s nem közösködik a fiatal kölykökkel, akik „kenik-vágják a témát". De mindig öröm újabb és újabb pedagógusoktól hallani, hogy „igen, én félre mertem tenni, hogy nagynak hittem magamat". És ha tudományról, azaz számítástudományról beszélünk, az új ismeret mércéje mindenképpen a nyilvánosság. Ebből az előremutató, nyílt versenyből a zárt rendszerek, egyszersmind a bezárkózó tanárok is mind kimaradnak.

Kovács Zoltán
(kovzol@math.u-szeged.hu)
tanársegéd a Szegedi Tudományegyetem Bolyai Intézetében az Analízis Tanszéken, matematikát és számítástechnikát tanít óraadóként a szegedi Radnóti Miklós Kísérleti Gimnáziumban.

LME-hírek

A Linux-felhasználók Magyarországi Egyesülete (LME) ebben az évben az Infoszféra Kft.-vel együttműködve 2002. november 9-én (szombaton) rendezi meg lassan már hagyományosnak tekinthető, nagyszabású GNU/Linux Szakmai Konferenciáját.

A konferencia Budapesten a VII. Kerület, Rákóczi út 90. szám alatt található Grand Hotel Hungária szállodában kerül megrendezésre, ahol az egyesület sok szeretettel vár minden érdeklődőt. Az IV. GNU/Linux Szakmai Konferencia célja, hogy lehetőséget nyújtson a magyar linuxos világ szereplőinek a bemutatkozásra. A bemutatásra kerülő előadásokat előzetesen szakmai zsűri ellenőrzi. A szervezők célja, hogy minden eddiginél színvonalasabb és nagyobb szabású rendezvényen láthassák vendégül az érdeklődőket.
A konferenciára elektronikus úton a
➲ http://konf2002.lme.linux.hu/ oldalon lehet jelentkezni.
A fenti oldalon a tervezett előadástémákkal kapcsolatban közvélemény-kutatást folytatnak, ahol bárki (célszerűen a konferencia leendő látogatói) a szervezők tudomására hozhatja, hogy a tervezett témák közül szerinte melyik lesz a legfigyelemreméltóbb.
Az előadások délelőtt fél 11-kor kezdődnek, és 2–3 csoportban zajlanak délután hat óráig, a következő tervezett időrendben:

Megnyitó	10^{00}
1. előadás	10^{30}–11^{15}
2. előadás	11^{35}–12^{20}
Ebédszünet	12^{30}-14^{00}
3. előadás	14^{15}–15^{00}
4. előadás	15^{30}–16^{15}
5. előadás	16^{35}–17^{20}
Zárszó	17^{45}
Zárás	18^{00}

A konferencia várható előadástémái:

- *Szalai Ferenc* (szferi@angel.elte.hu): Grid-rendszerek
- *Deim Ágoston* (ago@lsc.hu): Az embertől az államig (A nyílt forráskód haszna a kormányzatok számára; gazdasági és egyéb kérdések)
- *Varga Csaba Sándor* (guska@guska.hu): Az LME múltja és jövője (Bő lére eresztett számvetés: mi történt 1996 óta?)
- *Czakó Krisztián* (slapic@linux.co.hu): E-mail SPAM-védelem
- *Czakó Krisztián* (slapic@linux.co.hu): Biztonságos távmunka (IPSec VPN Linux segítségével)
- *Mátó Péter* (atya@andrews.hu): IDS
- *Németh László* (nemethl@gyorsposta.hu): Magyar Ispell/MySpell
- *Bodnár Csaba* (bocs@mclx.hu): A Caesar-kódtól az SSH-ig (A rejtjelezés rövid története)
- *Papp Dániel* (dani@mclx.hu): A Linux-alapú fürtözési megoldások áttekintése
- *Körmendy Domonkos* (doma@kormendy.hu): A PHP-GTK
- *Dr. Szentiványi Gábor* (szenti@suselinux.hu): Kufárok a bazárban, avagy a Linux dilemmái az üzleti életben
- *Fodor Orsolya* (fodorsi@axelero.hu): Weboldalkészítés Linux alatt (Multimédia és grafika témakörben)
- *Harka Győző* (carlos@gamma.ttk.pte.hu): A GNU/Linux rendszermag finomhangolása tűzfalakon
- *Milus János* (j.milus@chello.hu): Nagyrendszerek felépítése
- *Zahemszky Gábor* (zgabor@Picasso.Zahemszky.hu): A FreeBSD világa
- *Kadlecsik József* (kadlec@blackhole.kfki.hu): Csomagszűrő útválasztók
- *Scheidler Balázs* (bazsi@balabit.hu): Webkiszolgáló védelme határvédelmi eszközökkel
- *Kósa Barna* (barna_kosa@freemail.hu): Központosított felhasználókezelés GNU/Linux-környezetben
- *Deim Ágoston* és *Illés Márton*: UHU-kiszolgáló és tűzfal
- *Zelena Endre* (ezelena@lme.linux.hu): Open Source-tévhitek és cáfolatok
- *Zelena Endre* (ezelena@lme.linux.hu): Digitális aláírás és időbélyegzés: ajánlott levél elektronikusan?
- Egy könnyed hangvételű előadás az UHU-ról (Hogyan jutottak idáig, ügyfél, kiszolgáló, tűzfal, eLearning stb.) Az UHU által támogatott fejlesztésekről 45 percben (MPlayer, Gnome2-magyarítás, Pamir)

A konferencia előadásaiból nyomtatott kiadvány készül, amit a Konferencia vendégei belépéskor kapnak kézhez. A könyv várhatóan 150 oldalas lesz.
A konferencia után az elhangzott előadásokból, hozzászólásokból részletes Követőkiadvány készül, melyet az Üzleti díjat fizető vendégek nyomtatott formában megkapnak. A kiadványok később, a korábbi évek gyakorlatának megfelelően elektronikus levélben is hozzáférhetők lesznek.
A belépés LME-tagoknak ingyenes, feltéve, hogy előre befizetik a 2002. évi rendes tagdíjat (4000 Ft).
Nem tagok számára a belépőjegy ára: 1600 Ft+áfa (bruttó 2000 Ft); diákoknak, nyugdíjasoknak, katonáknak, tanároknak 50% kedvezményt biztosítunk, azaz a kedvezményes jegy ára: 800 Ft + áfa (bruttó 1000 Ft).
A korábbi évek gyakorlatának megfelelően most is lehetőség nyílik Üzleti belépő megvásárlására, amelynek összege 17 900 Ft + áfa (bruttó 22 375 Ft).

További információk a ➲ http://konf2002.lme.linux.hu/ oldalon találhatók.

Gibizer Tibor
(gibzo@linuxmania.hu)
újságíró, immár hét éve a Linux elkötelezett híve. Imádja a kutyákat, a kerékpározást és az autós csavargást.

UHU-Linux (Jóságpor)

Alig másfél évvel ezelőtt röppent fel a hír, hogy végre hazánknak is lesz saját Linuxa. Mára eljutottunk odáig, hogy megjelent a magyar nyelvű, magyar fejlesztésű, UHU-Linux névre hallgató operációs rendszer.

Sokakban felvetődik a kérdés, hogy mi végett van szükség hazai Linuxra, miért nem jók nekünk a már meglévő magyar nyelvi támogatásokat is tartalmazó egyéb operációs rendszerek? A gyenge válasz lenne, hogy van a németeknek, a franciáknak, sőt még a kínaiaknak is, akkor legyen nekünk is. Ezért, és hogy minél többet megtudhassunk az UHU-Linuxról, megkerestem ***Körmendi András***-t (andras@uhulinux.hu), az UHU-Linux fejlesztői csapat irányítóját. ***Gibizer Tibor*** (a továbbiakban Gibzo): Miért van szükségünk saját Linuxra, bár azt is kérdezhetném, mi a célja az UHU-Linuxnak?

A felhasználói szerződés elfogadása

Körmendi András: Az egyszerűség kedvéért megpróbálok pontokba szedve válaszolni. Annyi kitérőt engedj meg, hogy elmondhassam, ez az a kérdés, amit mindenki feltesz, és amire a legnehezebb tömören választ adni. A számtalan ok közül néhány fontosabb:

- Az egyéb operációs rendszereket használóknál a „nem fizetős" keresletet üldözi a BSA, nekik biztosan tudunk jól használható megoldásokat nyújtani.
- Közelítünk az Európai Unióhoz, ahol a nyílt forrású felület a kedvezményezett. Jelenleg jók az esélyeink, nincs az országnak lemaradása.
- A legtöbb állam elindította „nemzeti Linux" programját, ami az egymástól független fejlesztések ellenére a szabványok betartása miatt üzembiztos és biztonságos, ugyanakkor egymáshoz képest átjárható megoldásokat jelent.
- Az állami megrendelők számára a hazai fejlesztésű terjesztés biztosítja a felmerülő gondok azonnali megoldását, szemben egy nemzetközi cég kereskedelmi kirendeltségével, aminek – érthető módon – elsődleges célja nem lehet más, mint eladni.

A félreértések elkerülése végett: egyetlen külföldi céggel sincs gondunk, de a kereskedő kereskedő marad, és azt adja el, amit a fejlesztő kitalált, elkészített.
Mi ezzel szemben fejlesztők vagyunk, az a célunk, hogy a lehető legjobb programokat készítsük. Szélsőséges esetben akár naponta is képesek vagyunk frissíteni a csomagokat, hogy minél hibamentesebb, a felhasználói igényeknek megfelelő programokat tudjunk közreadni. Igaz, hogy a Linux a „népek olvasztótégelye", de büszkén vállaljuk, hogy kik vagyunk, és nekünk is sikerült elkészítenünk valamit, ami nem német, francia vagy amerikai, hanem magyar.

Gibzo: *Hosszú és fáradságos időszakot tudhattok magatok mögött. Most a jól megérdemelt pihenés időszaka következik vagy a hibajavításoké?*
András: Pihenésről szó sem lehet. Természetesen rendkívül boldogok voltunk, amikor kézbe vehettük az első kész korongokat, azonban azt is hozzá kell tennem, hogy ezzel az eddigi munkánknak csak töredékét mutattuk meg, de erről majd később árulok el részleteket. Visszatérve a kérdésedre, folyamatosan javítjuk a hibákat. Sajnos – mint azt éppen te rendkívül találóan megjegyezted – a PC lényegében egy kártyavár, ezért meglehetősen nehéz helyzetben vannak azok a fejlesztők, akik olyan rendszert szeretnének készíteni, ami mindenhol kifogástalanul működik. Tehát folyamatosan dolgozunk.
Gibzo: *Gondolom, elég komoly szervezet szükséges hozzá, hogy minden kihívásnak meg tudjatok felelni.*
András: Lényegében három fejlesztői „rétegről" beszélhetünk: vannak, akik főállásban csinálják, vannak velünk szorosan együtt dolgozók és akadnak alkalmi segítők is. A fejlesztés jelen szakaszában pedig mindenki, aki UHU-Linuxot használ, és megosztja velünk tapasztalatait.
Gibzo: *Hogyan kezdődött a fejlesztés, és miért pont UHU-Linux lett a neve?*
András: A hivatalos változat szerint linUx HUngary, de gondolom megoszthatok a Linuxvilág olvasóival egy titkot. Az egyik fejlesztőnk a névkeresés időszakában valamilyen, ma már homályba vesző apropóból elkiáltotta magát, hogy „Juhhúúú". Innen csak egy lépés volt az UHU.
A fejlesztés nagy lendülettel, komoly szembeszéllel kezdődött. Kezdetben nagyon kevesen hittek bennünk, igaz, mi is csak tapogatóztunk. Ma is felvetődik az a kérdés, hogy miért nem egy nagyobb változatra építettük az UHU-Linuxot? Természetesen ezt a megoldást is át kellett gondolni, azonban végül úgy döntöttünk, hogy ha hosszútávra tervezünk, inkább az alapokról induljunk el. Jelenleg úgy tűnik, jól döntöttünk. Amennyiben javítást vagy valamilyen igazítást kell végrehajtanunk, nem kell várni senkire, mi magunk elvégezhetjük belátásunk szerint.
Úgy látom, ma az a legfontosabb, hogy gyorsak és rendkívül rugalmasak legyünk ahhoz, hogy a munkánkat megbecsüljék, és minél szélesebb körben használják.
Gibzo: *Az UHU-Linux ügyfélváltozata mindenki számára szabadon elérhető az ➲ ftp.uhulinux.hu-ról, illetve azt is vállaltátok, hogy ingyen elküldítek a lemezt annak, aki a postai költségeket vállalja, tehát utánvéttel is beszerezhető. Így valóban elég sokan juthatnak hozzá. Kinek ajánlanád?*
András: Szívem szerint azt válaszolnám, hogy mindenkinek, de ez még csak az rc1-es változat, ami azt jelenti, hogy akadnak benne apróbb hibák. Ezek szerencsére nem befolyásolják a megbízhatóságot, tehát az UHU-Linux is legalább annyira biztonságos, mint az egyéb

Linuxok. Ellenben a számítógép különböző egységeinek életre keltése terén még akadnak feladatok, de mint már említettem, szinte naponta frissülünk. Érdemes az általad is említett FTP-kiszolgálónkat, illetve a honlapunkat ➲ http://www.uhulinux.hu figyelemmel kísérni, ahol minden változást igyekszünk közzétenni.

Fontos megjegyeznem, hogy az UHU-Linuxban elsősorban ugyanazokra az alkalmazásokra lehet számítani, mint a többiben, mivel arra sajnos nincs lehetőségünk, hogy a nagyobb programokat saját erőből készítsünk el. Talán egyszer ez sem lesz elképzelhetetlen. Jelenleg annyit tudok mondani, hogy az UHU-Linux ügyfélváltozatával azokat a felhasználói igényeket próbáltuk teljesíteni, ahol a Linuxot otthoni felhasználásra szánják vagy irodában szeretnék használni.

Gibzo: *Nem lehetett könnyű egy korongra zsúfolni mindent. A csomagok összeállításánál milyen szempontokat vettetek figyelembe?*

András: Az UHU-Linux egylemezes változatnak indult, de már most látszik, hogy ez nem tartható. Egyetlen CD-re csak azok az alkalmazások fértek el, amit az otthoni, esetleg néhány fős cégek igényelhetnek, gondolok itt elsősorban az irodai vagy a multimédiás alkalmazásokra.

Nagyon sok levelet kaptunk a fejlesztői csomagok hiánya miatt, viszont képtelenség volt mindent berakni. Egyelőre úgy néz ki, hogy előbb vagy utóbb két CD-s lesz az UHU-Linux. Az első korong önmagában is használható megoldást fog biztosítani a felhasználók számára, míg a második az egyéb kiegészítőket fogja tartalmazni.

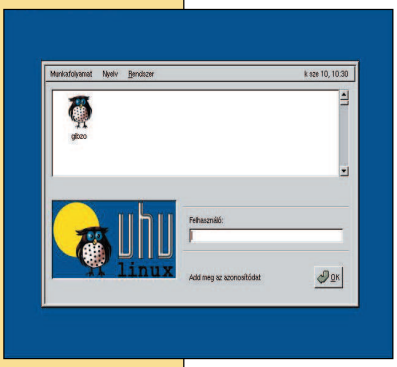
Gibzo: *Milyen saját fejlesztések találhatók az UHU-Linuxban? Tudom, erre lehet azt a választ adni, hogy az UHU-Linux maga is saját fejlesztés, viszont én kimondottan az általatok készített alkalmazásokra lennék kíváncsi.*

András: Valóban jól ráéreztél, alapjába véve az UHU-Linux mint terjesztés saját fejlesztés, hiszen nem valamelyik Linux-terjesztésre épül. Ezen belül is több olyan rész van, amit vagy kiegészítettünk, vagy újat írtunk belőle. Legszembetűnőbb a telepítő- és a beállítópanel, de számos olyan modul megtalálható benne, amit a hazai igényeknek megfelelően teljesen átdolgoztunk.

A válogatás természetesen tartalmaz néhány magyar fejlesztésű programot is, ezek máshol még nem részei a rendszernek. Ezt azért volt fontos megemlítenem, mert eddig, de a továbbiakban is erőnkhöz mérten szeretnénk támogatni a magyar fejlesztők által készített programokat.

Jelenleg a „támogatás” szó kétirányúságot jelent, hiszen mi is élvezzük mások munkájának gyümölcsét.

Több projektet is „támogatunk”. Ezek közül kettőt feltétlenül ki kell emelnem. Az mplayeres csapattal a legjobbkor találkoztunk, és azóta is felhőtlen a kapcsolatunk. Fejlesztői oldalról közelítjük meg a kérdéseket, egyik csapatnak sincs elrugaszkodott kérése, számíthatunk egymás segítségére. Úgy látom, hogy igazi csapatmunka alakult ki a két társaság között, ahol a „támogatás” szó nemcsak azt jelenti, hogy ezzel-azzal segítjük a másikat, hanem valóban örülünk egymás sikereinek – ez pénzzel nem mérhető. A Gnome honosítása – itt a Gnome2-re gondolok – teljesen új feladat. Ebben az esetben közvetlenül anyagiakkal tudtuk, illetve tudjuk gyorsítani a fordítási munkákat. Remélhetőleg sikerül a Gnome2 magyarázó szövegeit (help) is lefordítani, és attól a pillanattól kezdve nemcsak a menük, a hibaüzenetek lesznek magyarul olvashatók, hanem minden. Természetesen a folyamatos karbantartásra is figyelni kell, így adja magát, hogy hosszú távú kapcsolat alakuljon ki az UHU és a hazai Gnome-fordítók között. Úgy érzem, nagyon jó úton haladunk, itt is számíthatunk majd egymás segítségére.



Bejelentkezés a rendszerbe

Gibzo: *Milyen fejlesztési témákkal foglalkoztok még, illetve milyen egyéb terveitek vannak a közeljövőre nézvést?*

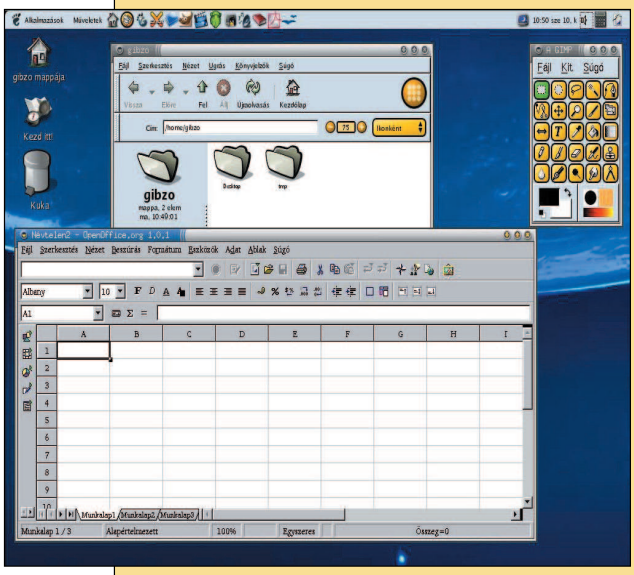
András: Három jól elkülöníthető irányt említhetek meg. Mindegyik külön-külön is megállja helyét, de terveink szerint hosszú távon jól illeszkedik majd az UHU-Linux projektekben.

- Több hazai fejlesztőcég egyesítette erőit és elkezdtük az UHU tűzfal-, valamint az UHU kiszolgálóváltozatának fejlesztését. Várhatóan még ebben az évben mindkettő kikerül az Internetre. A cél ugyanaz, mint az UHU- ügyfélváltozatnál: ingyenes letöltés, könnyű telepíthetőség, egyszerű beállítás és folyamatos terméktámogatás.
- Szeptemberben elindítunk egy UHU-Linux pályázatot, ahol neves támogatóinknak köszönhetően értékes jutalmakkal (notebook stb.) díjazzuk azokat, akik a legjobb pályaműveket küldik nekünk alkalmazás, Linux-oktatás, tanulmányírás és grafikus munkák témakörben. Mire ez a cikk megjelenik, a pályázattal kapcsolatos összes tudnivalót elérhetővé tesszük a weblapunkon ➲ http://www.uhulinux.hu).
- Hamarosan elindítunk egy – nem tudok rá jobb szót – „önképzést segítő” weboldalt, ahol alap-, és később remélhetőleg középszinten tanulhatnak azok, akik most kezdik az ismerkedést az UHU-val.

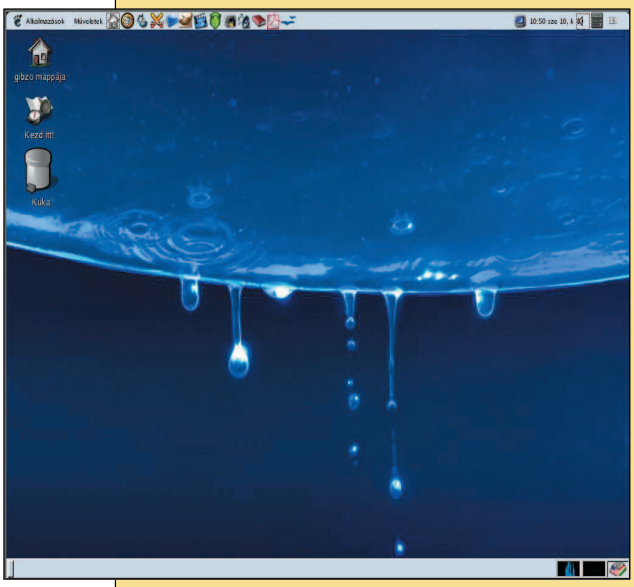
Gibzo: *Nemsokára a Linux minden területén megbízható, jól működő magyar megoldásokat tudtok nyújtani. Ez valóban a jövőbe vezető út, hiszen az összetett megoldások révén a kis- és közepes méretű vállalatok is bizalommal használhatják az UHU-Linuxot. Tudom, hogy pontos adatok nem állhatnak rendelkezésedre, de véleményed szerint mekkora lehet az UHU-Linux felhasználói tábora, mi a végső cél?*

András: Mennyien használnak UHU-Linuxot? Jó kérdés. Úgy augusztus elejétől érezni lehetett a felhasználók

érdeklődését, szeptemberben az rc1 kiadása után ez még markánsabban jelentkezett. Biztosnak látszik az UHU terjedése, sokan csak buzdítanak bennünket a további munkára, viszont egyre többen szeretnének tevőlegesen is segíteni. Az UHU-val foglalkozó



Az OpenOffice.org irodai csomag táblázatkezelője



A Gnome 2 alapértelmezett bejelentkezése

internetes oldalak száma is folyamatosan nő, tehát valami elindult, és ennek mi nagyon örülünk.
Mi a végcél? A Linux hazai térnyeréséhez el kell érni azt a „szükséges felhasználói tömeget”, amit már lehetetlen leállítani, és képes a saját fenntartására. Ehhez azonban nemcsak felhasználók, fejlesztők, magyar fejlesztésű terjesztés és hazai igényeket kielégítő alkalmazások kellenek, hanem teret kell hódítani a médiában, a piacon, be kell bizonyítani a közigazgatásnak, hogy életképes ez a „mások által nem annyira kedvelt” Linux. Ez azt jelenti, hogy ha tetszik, ha nem, a politikusokat is meg kell győzni. Sokan ellenzik ezt, azonban ez az a pont, ahol – szerintem – engedni kell. Állami megrendelések nélkül nehezen lehet bejutni az állami szektorba. Állami megrendeléseket pedig nem tudok elképzelni szakmai szempontból meggyőzött politikusok nélkül. Ezért nem panaszkodni kell, hanem makacsul menetelni előre, dolgozni, és lépésről lépésre elfoglalni azokat a területeket, amelyeket el lehet érni. Egyáltalán nem érzem, hogy le lennénk maradva. Az EU-csatlakozás közeledtével minden valószínűség szerint a nyílt forráskódú program lesz a jó irány, tehát nem szabad ölbe tett kézzel üldögélni, meg kell ragadni minden alkalmat, hogy a Linux-közösség felhívja magára a figyelmet.

Gibzo: *Köszönöm a beszélgetést, és további jó munkát kívánok.*

Az alábbiakban az UHU-Linux telepítését mutatjuk be tíz könnyű lépésben.

Telepítés

Az UHU-Linux telepítéséhez első teendőnk, hogy számítógépünk BIOS-ában lehetővé tegyük a CD-ről történő rendszerindítást, és természetesen helyezzük be a lemezt a meghajtóba.

1. lépés

A rendszer elindulását követően megjelenik a grafikus felület, és máris olvashatjuk a felhasználói szerződést, ahol az egerünket is beállíthatjuk.
Az esetek többségében az egeret önműködően felismeri, de az is előfordulhat, hogy nekünk kell kézzel beállítani, vagy ha ez sem segít, az
➲ ftp.uhulinux.hu címről tölthetjük le a legfrissebb egérkezelő csomagot, amivel biztosan működni fog.

2. lépés

Ki kell választanunk, hogy merevlemezünk melyik részére szeretnénk telepíteni az UHU-Linux rendszert. Ha teljesen üres merevlemezünk van, egyszerűen válasszuk a *Teljes merevlemez felhasználása* nyomógombot. Abban az esetben, ha már egyéb operációs rendszer telepítve van a merevlemezre, célszerű egy megfelelő programmal csökkenteni a meglévő rendszer méretét, és az így felszabadított helyre fogjuk telepíteni az UHU-Linuxot. Ez az előkészítő program lehet a *PartitionMagic*. A rendszer készítői a következő méretek beállítását javasolják:

- 2 GB a Linux rendszernek
- 256 MB a virtuális memória (swap) számára

Ebben az esetben a *Meglévő vagy üres partícióra* nyomógombot válasszuk.

3. lépés

A telepítés következő lépése a merevlemez-felosztás: egy táblázatból kell kiválasztanunk, hogy hova

szeretnénk telepíteni a rendszert. Itt tudunk módosítani a virtuális memória méretén is. Tudnunk kell, hogy a kiválasztott merevlemezrészen az összes adat el fog veszni. Az *OK* gombra kattintva kezdetét veszi a merevlemez formázása. A művelet befejezéséig várjunk türelemmel.

4. lépés

A *Programok telepítése* pontnál előre összeállított csomagkészletek közül választhatunk: *Alaprendszer; X grafikus felület; Gnome; Multimédia; Iroda; OpenOffice.org; Hálózati ügyfél; Játékok; Konzolos programok; KDE; Hálózati kiszolgáló; Fejlesztői programok.*

Akik nincsenek tisztában az olyan szavak jelentésével, mint a Gnome vagy az X grafikus felület, válaszszák az *Indulhat a telepítés* nyomógombot. A tapasztaltabb felhasználók és a részletesebb adatokra vágyók válasszák a *Beállítás egyénileg* gombot.

5. lépés

Amennyiben a teljes programválasztékot telepítjük, 1,6 GB szabad helyre van szükségünk. A telepítés időtartama számítógépünk kiépítésétől függ. Egy középkategóriás gépnél 15–20 perc alatt lezajlik. Először a kiválasztott programok kerülnek másolásra az előzőleg kijelölt merevlemezre, majd a csomagok beállítása következik, ami önműködően történik.

6. lépés

Ekkor kell eldöntenünk, hogy a rendszerbetöltőt hová szeretnénk helyezni. Ellentétben az általánosan megszokott LInuxLOader (LILO) helyett, az UHU-Linux a GRUB-ot használja, amelynek kezelése és beállítása a LILO-hoz hasonló módon történik. Az alapértelmezés szerint a GRUB a kiválasztott merevlemez rendszerindító területére (MBR) kerül, de választhatjuk az UHU-lemezrész indítórészét (boot) vagy kihagyhatjuk a rendszerbetöltő települését, ez utóbbi nem ajánlott. Az UHU-Linux indítólemez nélkül is használható, azonban azt ajánljuk, hogy a *Készít...* gombra kattintva hozzunk létre egyet, arra az esetre, ha használat közben véletlenül olyan maradandó károkat okoznánk rendszerünknek, amelynek következtében a rendszerbetöltő nem indulna el. A továbbhaladáshoz nyomjuk meg az *OK* gombot.

7. lépés

Linuxban a rendszerbeállításokat csak a rendszergazda tudja módosítani. A *Rendszergazda jelszó beállítása* alatt adjuk meg a jelszót, ami legalább hatkarakteres legyen; ügyeljünk rá, hogy a rendszer a kis- és nagybetűk között különbséget tesz, ne használjunk ékezetes karaktereket és szóközt. Ajánlatos kerülni az `y` és `z`, valamint a `0` karakterek használatát.

8. lépés

Az UHU-Linux mindennapos használatához biztonsági okokból nem ajánlatos rendszergazdai jogosultságokkal belépni, ezért szükségünk van egy egyszerű felhasználóra – ennek létrehozásában segít a *Felhasználói jelszó beállítás* oldal. Adjunk meg egy felhasználói nevet és a hozzá tartozó jelszót. A felhasználónév legfeljebb nyolc karakter hosszú lehet, és csak az angol ábécé kis- és nagybetűit, illetve számokat tartalmazhat.

9. lépés

A *Grafikus felület beállítása* oldalon grafikuskártyánk az esetek nagyobbik részében önműködően felismerésre kerül. Természetesen lehetőségünk

Az uhu control centerben fontos beállításokat végezhetünk

van, hogy grafikuskártyánk típusát és a használni kívánt színmélységet megváltoztassuk. A tökéletes grafikus felület eléréséhez képernyőnk beállítására is gondot kell fordítanunk. Ellenőrizhetjük beállításunk helyességét, majd az *OK* gombra kattintva befejeződik a telepítés.

10. lépés

A látvány talán ijesztőnek tűnhet az egyéb operációs rendszerekhez szokott szemeknek, hiszen a képernyő elsötétül, majd a rendszerindításkor lezajló folyamatokat szemlélhetjük. Ilyenkor karakteres felületet látunk, ahol magyar üzeneteket olvashatunk. Rövid várakozás után visszakapjuk a grafikus felületet, ahol a telepítéskor létrehozott felhasználóval beléphetünk az UHU-Linuxba, illetve választhatunk a KDE 3 és a Gnome 2 grafikus környezetek között.

Gibizer Tibor
(gibzo@linuxmania.hu)
újságíró, immár hét éve a Linux elkötelezett híve. Imádja a kutyákat, a kerékpározást és az autós csavargást.

Hogyan jutott zátonyra egy nyíltforrás-megegyezés

A Progress és NuSphere kontra MySQL AB a hanyag üzletkötésről, és nem a nyílt forráskód integritásáról szól.

Sokak szerint egy új, a bostoni szövetségi bíróságon folyamatban levő per „az első bírósági eljárás, amely a General Public License (GPL – általános közengedélyezés) érvényességét és alkalmazhatóságát elemzi". Na és? Vajon tényleg befolyással lesz ez a per a Linux-programozók jövőjére? Fontos-e ez a vita az olyan vállalatok számára, amelyek üzleti modelljeiket a nyílt forrású programokra alapozzák? Lesz-e a bírónak alkalma megbüntetni egy pimasz, a GNU régóta ismert szabályait megszegő programkészítő céget, és ezáltal megvédenie az OSS-ügyet, mint ahogy egyes OSS-ügyvédek felvetették? Sajnos valószínűleg nem. Igen, az ügy fontos. Igen, az általános vélekedés szerint ez vitathatatlanul a GPL első bírósági próbája. De nem szolgál előrejelzésként az OSS jövőjéről, mivel a vita egy rendkívül gyenge szerződés körül forog, hátterében a felek közti állandóan áttekinthetetlen és állandóan változó kapcsolattartással.
Nem lehet egy programozási nyelv kilátásait csupán egyetlen, az adott nyelven írt, rövid és rosszul dokumentált alkalmazás elemzésével előrevetíteni. És ebben az esetben az ügy alapját képező szerződés olyan messze elmarad az észszerű, korszerű programkezelési és -szabályozási gyakorlat normáitól, hogy egyáltalán nem helyezhető ugyanabba a fogalomkörbe. Minden bizonnyal értékesebbnek fog bizonyulni a „Programkezelési gyorstalpaló 101" (Software Product Management 101) és „Programszerződések kezdőknek" (Beginner Software Contracts) tanfolyamokhoz, mint az OSS-stratégiák finomításához.

Pillanatkép egy kisiklott vonatról

A történetet a nyilvánosság számára is elérhető bírósági perbeszédek tárják elénk. A pereskedő felek vitájának alapjául szolgáló szerződést a jól ismert MySQL OSS adatbázis finn szerzőinek vádcáfolatához csatoltan, az amerikai programkiadó, illetve -viszonteladó cég által indítványozott vád nyomán nyújtották be (tehát a szerződés és a felek különböző érvei, elektronikus levelei és eskü alatt tett írásbeli nyilatkozatai „nyílt forrásúak" a műszaki menedzserek, ügyvédek és oktatók számára, tanulmányozásra és az eljárások tökéletesítésére).
E cikk szerzőjének tulajdonába került a bírósági iratok között az az eredeti nemzetközi megegyezés, miszerint egy nyilvánosan jegyzett, régóta fennálló, massachusettsi telephelyű üzleti programkészítő cég viszonteladási jogot nyert egy fiatal, külföldi fejlesztőtől Finnországban. A csúf meglepetés: a két cég egy csupán kilenc bekezdésből álló szerződés alapján egyezett meg egy jelentős, nagy összegeket megmozgató alkuról.
A megegyezés összesen egy és egynegyed oldal hosszúságú volt!
A Progress Software nevű cég körülbelül 300 ezer amerikai dollár kifizetésébe egyezett bele egy új, (a Progress számára) ismeretlen iparrészlegben működő dinamikus külföldi cég részére – egy szóbeli megállapodás értékével egyezően. A MySQL AB finn cég egy rövid nyilatkozattal adta áldását arra, hogy fő terméke a massachussettsi viszonteladó tulajdonába kerüljön. A nyilatkozat szerint egy „későbbi" szerződés megkötésével a jövőben „összesen akár 2,5 millió dollár értékű áruforgalmat" fognak lebonyolítani. Az ebből származó vitából tisztán látszik, miért ráncolják homlokukat a tapasztalt üzletemberek (az ügyvédeket is beleértve) a derűlátó „csak bízzunk egymásban, és majd később megbeszéljük az alkut és a részleteket" elképzelés hallatán.
Hogy mi a gond a tömörséggel és bizalommal? Nézzük így a dolgot: miért végezzünk műtétet, mielőtt röntgent készítenénk vagy megvizsgálnánk a beteg kórtörténetét? Miért ne ugorjunk fejest egy ismeretlen folyóba? Magunknak és másoknak is kárt okozhatunk egy jelentős programkezdeményezés indításával – legyen az OSS vagy szabadalmaztatott jellegű –, ha azt az alapszabályok meghatározása nélkül tesszük. Itt is pontosan ez történt.
A legtöbb szerződés egyik célja hasonló az adatfeldolgozás alapelveihez: teljesítményértékelés, kipróbálás és a színvonal biztosítása. Ebben az esetben az elküldött kód töredékes volt. Azaz túl korán és túlságosan sok műveletet alapoztak egy hiányos „megegyezésre".

Fülsüketítő, halálos csend

Mi az, ami kimaradt ebből a rövid és végső soron keserű szerződésből? Csupáncsak a legtöbb programok kapcsán létrejött szerződésben megtalálható feltétel és kikötés nagy része. Íme néhány dolog a sok közül, ami nyilvánvalóan hiányzik:

1. Mikorra készülne el a tervezett „későbbi, az előzőt felváltó szerződés"?
2. Az üzleti feltételeket illetően milyen jellemzők között zajlott?
3. A technikai háttér biztosításához milyen szintű szolgáltatások lennének szükségesek, illetve milyeneket nyújtanának? Mit értenek „vállalati szintű támogatás" és a „fennálló elektronikus segélynyújtási csatornák" alatt?
4. Kik lennének a vállalatok közti kapcsolattartás lebonyolítására kijelölt képviselők?
5. Mit jelent, hogy egy vállalat partnerét fő védjegyének „méltányos használati" jogával ruházza fel, ahogyan azt a MySQL AB a fentiek szerint tette?
   Ez gyakorlatilag milyen eseteket hagyna jóvá, és melyeket zárná ki?
6. Milyen folyamatos termékfejlesztési szolgáltatásokat biztosítana az eredeti program készítője?
7. Hogyan oldanák meg vagy bírálnák el szükség esetén a vitás kérdéseket?
8. Ha a vita az egyik fél hibájából adódik, az köteles-e kártérítést fizetni, illetve az eljárás költségeit megtéríteni?
9. Mire jó kihagyni az összes olyan sokat becsmérelt, típusos vagy szabványos óvintézkedést, amelyek éppen azért szerepelnek a legtöbb szerződésben, mert

lehetővé teszik azok érvényesítését és a viták elkerülését?

Tanuljunk mások kárából: úgy kódoljuk szerződéseinket, mintha programok lennének.

A legtöbb korszerű, tapasztalt programkészítő vállalat tisztában van azzal a számtalan kérdéssel, ami felmerülhet és fel is merül egy programforgalmazási megegyezés lebonyolítása során. Először tervezetet készítenek az egyezségről (például egy „szerződési feltételek" lista formájában vagy vázlattal), ezután azt „kódolják" (azaz egy előzetes szerződést írnak), majd kipróbálják és leírják a megállapodásokat (azaz tárgyalásokat folytatnak és finomítanak az alapszerződésen, valamint megírják és átvizsgálják a szükséges iratokat), ugyanúgy, ahogyan azt alkalmazásaikkal teszik. Sok programkészítési projektben például részletesen, előre megnevezik a „felhasználói követelményeket". A szóban forgó megállapodásból nyilvánvalóan hiányzott egy összesített „feltétellista" vagy „összefoglaló jegyzet a megállapodásról", amely az egyezség alapjául szolgálhatott volna.

A legtöbb alkalmazás keresztülmegy egy, a programozó kollégák által végzett minőségi ellenőrzésen. Egyes összetett programozási környezetekben automatizált kódkipróbáló eszközöket is bevetnek. Ezt a szerződést vélhetően egyetlen ember vagy legalábbis egy nagyon kis létszámú csapat keze munkájaként bocsátották útjára.

Példa a szegényes kódra

Megegyezés

Az alábbi feltételek a MySQL forgalomba hozatalát szabályozzák a GNU General Public License (GPL), valamint a Progress Software Corporation, az alábbiakban PSC, és a TCX DataConsult AB, MySQL AB, Monty Widenius és David Axmark, az alábbiakban MySQL AB közötti megegyezés értelmében.

1. A hivatalos bejelentés napján a PSC átutalja az első 74 167 dolláros összeget (104 167 dollár – 30 000 dollár előleg) a MySQL AB számára. Ez azon döntésünk bejelentésével együtt, mely szerint vállalati szintű támogatást nyújtunk a MySQL-nek, és ellátjuk egy megvételre kész termékcsomaggal, óriási mértékben meg fogja növelni a MySQL piaci hitelét, növelve hatásunkat a piaci elemzőkre, a piaci sajtóra stb. Ezt a lépést egy összesen legfeljebb 2,5 millió dollárig terjedő kifizetés első részleteként jelentenénk be, és legalább augusztus végéig folytatni fogjuk a 104 167 dolláros átutalásokat. Ezt követően az átutalásokat és egyéb intézkedéseket egy további megegyezésünk fogja szabályozni, amely megegyezést a pénzátutalások folytatása előtt még véglegesítenünk kell.
2. A MySQL AB felülvizsgálati joggal rendelkezik minden, a PSC és a MySQL AB kapcsolatára vonatkozó nyilvános adattal kapcsolatban annak biztosítása érdekében, hogy a PSC nem tesz véletlenül olyan kijelentést, ami korlátozhatja a MySQL AB esetleges jövőbeli lehetőségeit vagy zavart kelthet. A MySQL AB-nak lesz jogában jóváhagyni a PSC és a MySQL AB jelen és jövőbeli kapcsolatára vonatkozó összes sajtóközleményt. A MySQL AB ezennel kijelenti, hogy beleegyezését oktalanul nem fogja megtagadni.
3. A hirdetmény napján a TCX DataConsult AB, a MySQL AB, Monty Widenius és David Axmark a GPL alá helyezi a MySQL 3.23 változatot (a jelenlegi változatot).
4. A MySQL AB elismeri, hogy ennek a szerződésnek az értelmében a PSC rendelkezik a MySQL GPL licensz alatti eladási és terjesztési jogával a bejelentés napjától kezdődően.
5. A bejelentés napjától kezdődően a PSC-nek joga van forgalmazói áron nyomtatott MySQL-leírást vásárolnia a kiválasztott MySQL-leíráskiadótól. A PSC beleegyezik, hogy ezt a dokumentációt csakis a MySQL egy-egy példányával vagy egyéb számottevő érték hozzáadásával értékesíti. A PSC-nek joga van bármilyen szükséges leírást nyomtatnia magának abban az esetben, ha nincs semmiféle kizárólagos dokumentációforrás.
6. A PSC-nek joga van a „MySQL" védjegy méltányos használatához, beleértve a „NuSphere MySQL", az „enhanced „MySQL" és a „Rocket MySQL" kombinált védjegyekre kiterjedő bejegyzési és használati jogot.
7. A MySQL AB ötnapos, közös megállapodás alapján ütemezett, térítésmentes szakmai oktatást biztosít a PSC alkalmazottainak. A PSC feltételezi, hogy e szakmai gyakorlat előtt minden alkalmazottja elolvassa a MySQL kézikönyvet és ***Paul Dubois*** könyvét. A szakmai gyakorlaton MySQL AB-alkalmazottak is részt vehetnek.
8. A MySQL AB háttértámogatást nyújt a PSC által nyújtott segélyszolgáltatások kiegészítésére legalább augusztusig, a fennálló elektronikus segélynyújtási csatornákon keresztül. Ezt követően a MySQL AB még két évig háttértámogatást nyújt a PSC-nek egy későbbi megegyezés szerint, de megállapodás hiányában legfeljebb 150 dolláros óradíj fejében, minden megkezdett 15 perc után.
9. Ez a szerződés két vagy több példányban érvényes, ezek mindegyike eredetinek minősül, de együttesen mind egy és ugyanazon egyezményt képezik.

Ha ez a szerződés elfogadható az Ön számára, kérjük, írja alá és küldje vissza nekünk, hogy minél hamarabb elindíthassuk a MySQL-t mint egy „valódi nyílt forrású" adatbázist egy megbízható harmadik fél támogatásával, mielőtt a versenytársak elorroznák vezető helyét!

Progress Software Corporation

Patrick Lannigan,
Britt Johnston

Dörzsölt programozók programjaikat hibaüzeneti szolgáltatásokkal látják el. Ebből a ködös megegyezésből hiányzott a szokásos „szerződésszegési tájékoztatás, illetve az áthágás helyrehozatalának lehetőségét” biztosító óvintézkedés.
A tapasztalt kódolók fejállományokat (header) és egyéb technikai jellegű leírást is beépítenek a munkájukba a későbbi módosítást és hibakeresést megkönnyítendő.
A programkészítési üzleti ügyek során határozzuk meg pontosan a gyártó és a forgalmazó közötti kapcsolattartás lebonyolításának módjait. Nevezzük meg rögtön az elején azokat a személyeket, akik felhatalmazást kapnak a másik szervezet kinevezett személye vagy személyei felé üzleti utasítások, ellenvetések, illetve javaslatok közvetítéséhez.

Rémisztő képek: hajók az éjszakában

A szerződés tömörsége következtében elképzelhető, hogy a felek olyan jogi kérdéseket vetnek fel, amelyek megzavarják az ügyet, de legalábbis késleltetik a kimenetelét. Ne feledjük, milyen lassan forognak az igazságügyi rendszer fogaskerekei az Egyesült Államokban is. Minden bizonnyal csalódniuk kell a GNU-modell törvényes elismertetésében reménykedő OSS-híveknek: mindkét perben álló fél felvetett már az OSS-ügytől független jogi kérdéseket. A MySQL AB például már (február 28-án) nyert egy részleges bírósági végzést a Progress, illetve annak fiatal leányvállalata, a NuSphere ellen, de nem a GPL megerősítésének terén, hanem védjegyezési ügyben. A szövetségi bíró túlságosan bizonytalannak találta a GPL-ügyet ahhoz, hogy már a bírósági eljárás ezen korai, áttekintő szakaszában döntést hozzon róla. Vagy vegyük például a „kölcsönös vétség” jogi tantételét. Egy szerződés néha amiatt nem emelkedik jogerőre, mert a két fél óhatatlanul is eltérő, de különkülön ésszerű módon értelmezi a megegyezésben tett megállapodásokat és feltételeket. Ennek klasszikus példája egy hasonló nemzetközi félreértés.

Ha Rómába utazunk, tervezzünk előre

A történet elhamarkodottságát nemzetközi háttere még jobban kihangsúlyozza. A nemzetközi üzletkötések fokozott megfontolást és részletesebb feltételi leírásokat érdemelnek, mint ahogyan a több országban is használatos alkalmazások esetén több moduláris képernyőkiíratásra és (az ázsiai karakterkészletek miatt) két bites kódra van szükség, amelyekkel megfelelnek több különböző operációs rendszer iterációinak és egyéb fondorlatos kódolásnak.
A külföldi vállalatokkal való üzletek külön odafigyelést igényelnek. Számos külföldi vállalat például vitás esetekben szívesebben választja (vagy éppen ragaszkodik hozzá) a kérdés döntőbírával való eldöntését, ami egyrészt az erős kulturális hagyományból ered, másrészt abból a meggondolásból, hogy így elkerülhető az Amerikában hírhedten szokásos elhamarkodott, elnyúló és drága pereskedés (ebben az adott esetben a pereskedő felek 73 különböző bírósági előterjesztést nyújtottak be a per első kilenc hónapjában, és az ügy végeláthatatlannak tűnik).
A világjáró utazók indulás előtt gondoskodnak a tolmácsokról, utánpótlást biztosítanak és tájékozódnak a helyi kapcsolattartási protokollokról. A nemzetközi szerződések esetében sok cég hasonló kiegészítő intézkedéseket tesz. Előre megállapodnak a terméktervezési együttműködés minimális mértékében, szerződésben kötelezik el magukat egymás székhelyeinek meglátogatására és a jelentős világkereskedelmi kiállításokon való találkozásokra, és egyéb szerződésbeli „ragasztókódot” építenek be kapcsolataik finomítása érdekében. A józan ész azt diktálja, hogy készítsünk térképet, mielőtt ismeretlen tájakra merészkedünk. Ebben az esetben a felek eltévedtek, és a bíróságon találták magukat, ennek minden velejárójával: forgalmazási csapásokkal, hatalmas pereskedési számlákkal és a termék bizonytalan jövőképével együtt.

Mire gondoljunk, mit tegyünk?

A Progresst és a NuSphere-t az OSS-közösség egyes tagjai arra a valós, de nem teljes történetre hivatkozva támadták, miszerint a MySQL-kódot megváltoztatva, és azt utána szabadalmazott felhasználási szerződéssel nem pedig a GPL-lel, vagy valami egyéb OSS felhasználási szerződéssel hozták forgalomba. Való igaz, a NuSphere módosította a GPL használati modelljét, és ezzel az ő szemszögükből ítélve csak egy átmeneti figyelmetlenség következményét tették jóvá. De ez nem a teljes történet.
A perbeszédek alapján másik oldalról is nézhetjük: inkább a Progress érdemel bírálatot, amiért valamelyik jöttment termékmenedzsere, vélhetően a vezetői tanács és más kollégákkal való egyeztetés nélkül, egy rosszul dokumentált szerződést hozhatott létre. Büntetése legyen egy felhasználási szerződést tanító tanfolyamon való kötelező részvétel. Esetleg enyhítsenek az ítéleten a forgalmazásra sürgető piaci versenyre való tekintettel. Utána pedig biztosak lehetünk benne, hogy legközelebb mindkét vállalat hagyományos, következetes és kielégítő programkészítési szerződéseket fog alkalmazni, tanulván az ügyvédekre költött nagy összegekből, valamint az idő-, a vállalatvezetési energiákban és a vevőkörben bekövetkezett veszteségekből. A Progress-NuSphere-MySQL per végül lehet, hogy csupán egy újabb fejezet lesz azon vállalatok vaskos könyvében, amelyek megfelelő célzás nélkül „nyitottak tüzet”.

Linux Journal 2002. augusztus, 100. szám

Henry W. (Hank) Jones
22 éves programtanácsadó, menedzser és ügyvéd, a UC Berkeley Extension programengedélyezési workshop alapítója és vezetője, aki már több mint 75 programkészítő céggel dolgozott.

Nincs több korlát az internetes rádiózás kivégzése előtt

Mint a legtöbb havi magazin, mi is két-három hónapos átfutási idővel dolgozunk. Ezért nagy körültekintéssel olyan témát szoktunk választani a sajtószemleként, ami nem csupán időszerű, de nagy valószínűséggel három hónappal később is érdemlegeset tud még mondani. Olykor előfordul azonban, hogy a hírek áramlása 180 fokos fordulatot vesz. A hír időszerű marad ugyan, de a tartalma teljesen megváltozik. Ebben a hónapban is ez történt, és sajnálattal kell közölnünk, hogy az elmúlt hónapban jó hírként beharangozott esemény, miszerint a Szerzői jogok jegyzéke (Register of Copyrights) és a Librarian of Congress elutasította a CARP azon ajánlásait, amelyek szinte az összes egyesült államokbeli élő webes műsorszórásra veszélyt jelentettek, mostanra tévesnek bizonyult. A korábbi határozat, úgy tűnik, nem volt más, mint az internetes rádiózás kivégzésének időleges felfüggesztése. A hír akkor így szólt:

„2002. május 21-én a Librarian of Congress a Szerzői jogok jegyzékének ajánlásából kiindulva kiadott egy határozatot, amelyben visszautasítja a bizottság internetes rádiózásra vonatkozó, díjszabással és a fizetési határidővel kapcsolatos előterjesztését. Ilyen esetekben a törvény előírja, hogy a Librariannak a bizottság javaslatának visszautasításáról hozott végső döntését 30 napon belül ki kell adnia. A határozat kiadásának határideje tehát 2002. június 20."

Nem mi voltunk az egyetlenek, akik ezt a határozatot az internetes rádiózás szempontjából jó hírként tártuk a nyilvánosság elé. Szinte valamennyi elemző és szerkesztő így látta. Az egyedüli kivétel *Jonathan Peterson* volt, a Way.Nu-tól:

„Bár jó hír a webes műsorszóróknak, azért én egyelőre nem bontanék pezsgőt. Mérget vehetünk rá, hogy az RIAA és a lemezkiadók már most tárgyalási időpontokkal töltik tele a kongresszusi döntéshozók határidőnaplóit, hogy ezáltal is nyomást gyakoroljanak."

Igaza lett. Amikor elérkezett június 20-a, vagyis a Librarian of Congress webes műsorszórásra vonatkozó végső határozatának kibocsátása (➲ http://www.copyright.gov/carp/webcasting_rates_final.html), a megdöbbenés nem maradt el. Mintha a dokumentum az egy hónappal korábbi határozatban lévő kibúvón csúszott volna keresztül. A kezdő mondat így szól: „A Librarian of Congress elfogadta a Szerzői jogok jegyzékének ajánlását, és visszautasítja a CARP azon díjszabásokkal és fizetési határidőkkel kapcsolatos határozatát, ami a törvényesen működő, nem előfizetéses szolgáltatásokra vonatkozik, amelyek nyilvánosan hangfelvételeket játszanak le digitális audioátvitellel („webcasting") az Egyesült Államok Törvénykönyvének 17-es számú bejegyzésének 114 paragrafusa értelmében; valamint azokra a törvényes szolgáltatásokra vonatkozik, amelyek ideiglenes hangfelvételeket készítenek hangfelvételek lejátszásának céljából az Egyesült Államok Törvénykönyvének 17-es számú bejegyzésének 112-es paragrafusa értelmében." (a kiemelések tőlem származnak). A 114-es paragrafus az 1995-ös Digital Performance Right in Sound Recordings (A hangfelvételek digitális felhasználásával kapcsolatos jogok) nevű határozatra utal. A 112-es paragrafus pedig az 1998-as Digital Millenium Copyright nevű határozatra vonatkozik, vagyis a díjszabásra és a határidőre vonatkozó kitétel kivételével a CARP-határozat érvényben maradt. Az utóbbiak a kizárólag internetalapú állomások esetében egy adott hangfelvételre, egy hallgatóra jutó sugárzási egységre (stream) vonatkoztatva 0,0014 dollárról 0,0007 dollárra változtak. A nem kereskedelmi szimultán sugárzások 0,0002 dolláron maradtak, a nem kereskedelmi oldal csatornába első adások 0,0014 dollárról, szintén 0,0007 dollárra csökkentek. A „lejátszási díjak" kilenc százalékról 8,8 százalékra csökkentek. Az üzleti alapokon álló szolgáltatásokra (háttérzene, például Muzak) kiszabott minimális díj 500 dollárról 10 000 dollárra emelkedett.

A díjak 1998. október 28-ra visszamenő hatállyal érvényesek. Ez azt jelenti, hogy egy óránként kétezer hallgatóval rendelkező adó, amely óránként 18 dalt játszik le, 0,0007 dolláros dalonkénti, hallgatónkénti és óránkénti díjjal számolva naponta 604,80, míg évente 220 903,20 dollár tartozást hoz össze:

- 18 × $.0007 = $.0126 hallgatónként, óránként
- $.0126 × 2000 = $ 25,20 2000 hallgatóra számolva egy óra alatt
- $.31752 × 24 = $604,80 naponta
- $2,419655 × 365,25 = $220 903,20 évente

A Digitally Imported Radio (➲ http://www.digitallyimported.com) nevű adó, amely BSD-kiszolgálókról sugároz, és kimutatásait Linux-alapú programokkal készíti, átlagban több mint négyezer hallgatóval rendelkezik óránként a trance csatornán. A RadioParadise, amely linuxos alapon sugároz, és kizárólag nyílt forrású programokra épül, naponta körülbelül ezer hallgatóval bír. A RadioParadise ugyanarra az üzleti modellre épül, mint a közszolgálati rádió és tévé: a hallgatói hozzájárulásra. Ez mintegy háromezer dollárt jelent havonta, ami alig fedezi az üzembetartás költségeit. Számítsuk mindezt vissza 1998-ig, és máris látjuk, hogy egyes adók sok százezer dolláros tartozást halmoztak fel.

Bárhogyan is magyarázzák, a Librarian határozata olyan fondorlatnak tűnik, ami csődbe viszi az internetes rádiózás úttörőit. E cikk írásának idejére máris számtalan állomás kényszerült beszüntetni a tevékenységét, csakhogy sokan tovább folytatják, amit elkezdtek, és harcolnak a határozat ellen. A friss híreket a ➲ http://www.linuxjournal.com oldalon folyamatosan közöljük.

Linux Journal 2002. szeptember, 101. szám

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

Tudod-e?

1. A projekt fedőneve „Green Project” volt, és nem tervezték, hogy új programozási nyelvet hoznak létre. A „Green Team” a véletlennek köszönhetően alkotott meg egy új nyelvet, aminek az Oak nevet adták – a szerző irodája előtt álló tölgyfa után. Később, 1995-ben új nevet kapott. Melyik nyelvről van szó?
2. Mi a különbség az étkező filozófus és az ivó filozófus kérdése között?
3. Mi a különbség az && és az és között a Perl nyelvben? Melyik volt először?
4. A nyelv eredetijét *Aho*, *Weinberger* és *Kernighan* készítette 1977-ben az AT&&T Bell Labsnál. Ezt a feldolgozónyelvet a szerzők után nevezték el. Mi is a neve?
5. A nyelv készítője, amikor művét elküldte az Internet Software Consortium számára, az awk és a sed nyelv felváltójának nevezte. Melyik nyelv ez?
6. A szóban forgó programozási nyelvhez – szerzője szerint – egy kellőképpen rövid, egyedi és titokzatos név illik. A szerző nagy rajongója BCC hetvenes években játszott vígjátéksorozatának, amelyben többek között *Graham Chapman*, *John Cleese* és *Eric Idle* játszott. Nem a programozási nyelv nevét kérdezzük, mert az túlságosan könnyű volna. A kérdés inkább a következő: mi annak a dalnak a címe, melyet *Arthur Ewing* és éneklő egerei adtak elő a második részben?
7. *John McCarthy* nem volt elégedett az IPL-lel (listafeldolgozó nyelv a Rand Corporation Johnniac nevű gépére), ezért ugyanennek a gépnek a számára egy új nyelvet fejlesztett ki. 1960-ban megjelent egy írása, amely olyan jelentőségű a programozásban, mint Euklidész művei a geometriában. Melyik programozási nyelvről van szó?
8. Ki írta, hogy a „go to utasítás ártalmas”, és a következők közül mely nyelvek esetén vették ezt figyelembe: C, Perl, Python, Intercal?
9. „…Ha mégis hiba van a kódban, szíves örömest fizetek 20 dollár 48 centet a megtalálójának. Ez pontosan kétszer annyi, mint az eddigi, és terveim szerint minden évben megkétszerezem az összeget. Láthatják, biztos vagyok a dolgomban.” Ki és miről írta e sorokat? Segítség: a nyelv jelenlegi kiadása a 3,14159 (pi).
10. A LiveJournal.com újságírója, *Cassandra Claire* a következő „titkos naplót” szerezte meg. Melyik Gyűrűk ura-beli szereplő naplója ez? „Kiábrándultam a palantirfiúból. Nem hajlandó fényképet küldeni magáról, kivéve egy nagyon nagy szemgolyóról. Azt mondja, hogy félénk, de gyanítom, kövér vagy szőrös. Sok rosszat hallottam a palantirkapcsolatokról. Lehet, hogy szüneteltetni kellene egy kicsit”.

Válaszok

1. A Javáról.
2. Az étkező filozófusok kérdésében a filozófusnak annak érdekében, hogy enni tudjon, minden olyan villát meg kell szerezni, melyet megosztott a szomszédjaival. Az ivó filozófusok kérdésében viszont a filozófusnak csak néhány ilyen ünvegt kell megszerezni, hogy egy ital kevelhessen.
3. Csak kiértékelési sorrendiségükben különböznek: az és ugyanúgy működik, mint az &&, de az && előbb hajtódik végre, mint az és. Az és-t csak a Perl 5-tel vezették be.
4. awk
5. Perl
6. „The Bells of St. Mary’s”. A tévéműsor címe természetesen a Monty Python’s Flying Circus.
7. LISt Processing – LISP.
8. *Edsger W. Dijkstra* 1968-ban. A Python és az Intercal nem tartalmaz gotot, bár az Intercal megengedi a COME FROM használatát.
9. *Donald Knuth* arról, hogy mennyi pénzt ajánl fel annak, aki hibát talál a TeX-ben. Véletlenül a Metafont jelenlegi kiadásának száma 2.718 (e).
10. Fehér Szaruman

Linux Journal 2002. szeptember, 101. szám

Sumit Dhar

Ők mondták

A műszaki fogalmak „kreolizált” nyelve felé haladunk. Az ötlet a nyelvészetből való, mégpedig *Steven Pinker* munkájából, amelyben leírja, hogy különböző nyelvű felnőttek huzamosan egy helyen tartózkodva egy tört, darabos nyelvet hoznak létre, amelyet a nyelvészek pidginnek neveznek. A következő nemzedék által ez a nyelv összetettebbé válik, amelyből egy igazi nyelv alakul ki, ezt hívjuk kreolnak. Hasonlóképpen használják a mai gyerekek a műszaki megoldásokat – mi, felnőttek csak a pidgint alkottuk meg. *(Douglas Adams)*

A Perl az Internet jiddise. *(Yoz Graehme)*

Az érdeklődés a nyilvános szellemi tulajdon iránt akkor szűnt meg teljesen, amikor Kansas Cityben egy íróasztalon megszületett egy egér rajzfilmfigura. Az egeret mélyhűtőbe tették. Nem porlad el. Halhatatlan. Hűtéstechnikailag konzervált.... Az ókori Rómában úgy hitték, a hanyatlás jele, hogy Caligula a saját lovát tette meg szenátornak. Ennek ellenére a mai Egyesült Államok szenátusában van egy szenátor, aki egy egér rajzfilmfigura. *(Bruce Sterling)*

Linux Journal 2002. szeptember, 101. szám

A nyílt spektrum lendületet vesz

Ha bizonyítékot keresünk arra, hogy igény van vállalkozó kedvre és friss innovációra, gondoljunk a wardrivingra és warwalkingra. Egyiknek sincs semmi köze a hagyományos értelemben vett háborúhoz (war), bár mindkettőt a War Games című film ihlette. A WAR tulajdonképpen betűszót rejt magában: vezeték nélküli elérési felderítés (wireless access reconnaisance).

A vezeték nélküli elérhetőség felderítése jelenleg azokra a betyárokra (hacker) korlátozódik, akik sávszélességüket nyilvánosan elérhető helyeken, közintézményekben osztják meg, és azokra a hozzáértő hordozhatógép-használókra, akik kávézókat, könyvesboltokat és padokat keresnek, csakhogy ezt a sávszélességet használhassák.

A warwalkingnak két, a népszerű trendekre érvényes ismertetőjegye is van:

1. a betyárok azon serénykednek, hogy a hátterét ingyenessé és nyitottá tegyék,
2. a törvényhozók és az érintett vállalatok érdekeik szerint vagy ellene, vagy mellette foglalnak állást.

A támogatók bevezettek egy meglehetősen tág fogalmat: a nyitott spektrumot, ennek célja, hogy kiszélesítse azokat a kis számú csatornákat, amelyek alkalmasak a vezeték nélküli hálózati elérésre (WiFi vagy 802.11b). A mögöttes gondolat az, hogy tartalékolják (ahelyett, hogy kiárvereznék – a jelenlegi FCC-alapértelmezés) a spektrum azon szeletét, amely jogosítványok nélkül kerül felhasználásra.

A politikai hang szerepét *Johna Markey*, Massachussets állambeli képviselő vállalta fel azáltal, hogy a nyitott spektrumot bevezette a jogalkotásba. Egy nemrégiben elhangzott beszédéből idézve:

> „A parlament törvényhozásában a *Spektrum szokásjog* megalkotását akartam elősegíteni. A csúcstechnológiai termelők, a vállalkozók és a *kölykök a garázsban*-ként ismert fiúk sokkal erősebben ki tudnák használni a vezeték nélküli kapcsolattartásban rejlő lehetőségeket, ha a jogosítvány nélküli sávhasználatot kellően kiszélesítenék a közönség számára."

A törvényjavaslat szerint az FCC-nek két sávot kellene elkülönítenie közhasználatra, amelyek nyitottak a hatóságilag nem engedélyezett használatra. Az egyik egy 20 MHz-es összefüggő sáv lenne 2 GHz alatt, míg a másik 300–500 MHz 2 és 6 GHz között. Egy ilyen nyilvános elkülönítés nyitott felület létrejöttét segítheti elő, ami teret biztosíthatna az újításoknak, a vállalkozási tevékenységeknek és a tömegkommunikációnak. Ugyanakkor az ellen is küzdene, hogy a sávok túlságosan kevés szolgáltató kezében maradjanak. Múlt novemberben *Kevin Werbach*, korábbi ipari elemző és az FCC új technológiai kódexének tanácsadója az EdVenture Holding részéről nyílt levelet intézett *Michael Powell*-hez, az FCC elnökéhez. Ezekkel a sorokkal zárta gondolatait:

„Ön október 23-i sajtótájékoztatóján felesküdött, hogy erőteljesen támogatni fogja a sávok hatóságilag nem engedélyezett használatát, ahol csak elfogadható." Bár ez a jóváhagyás üdvözlendő, korántsem elég. A bizottságnak cselekvő módon kell megújítania a spektrumra vonatkozó üzletpolitikáját, hogy ne hátráltassa, hanem elősegítse az újítások létrejöttét. A Linux-pártiak talán képviselőtársaikat is arra akarják bátorítani, hogy ugyanezt tegyék.

Doc Searls

Linux Journal 2002. szeptember, 101. szám

Elvtársi üdvözlet!

„A kulturális érintkezés súlyos felelősséget hordoz magában: program a népek számára. Rendkívüli ár: 139,9 dollár"

„Kulturális kapcsolattartás: a Kínai 2000-rel rendelkezők bírnak a legnagyobb versenyteljesítménnyel. A műszaki hatalom a népé. Rendkívüli ár: 139,9 dollár"

Megjegyzés: a fenti árak hongkongi dollárban értendők. Ennek amerikai megfelelője körülbelül 18 dollárt tesz ki. Köszönet a CultureComnak (➲ http://www.culturecom.com.hk) a képekért és *Michael Fowler*-nek a fordításért.

Linux Journal 2002. szeptember

Mindenki üdvözli a nagyszerű emberek operációs rendszerét!

Rugalmas géptelep? Akkor openMosix!

Korábbi számainkban már olvashattak linuxos telepekről. Most egy másfajta megvalósítással ismerkedhetnek meg.

Az openMosix olyan rendszermag-kiterjesztés, ami telepek (cluster) építését teszi lehetővé. Hogy miért is jó? Miért pont ezt válasszuk? Remélem, ez az írás egyértelmű választ ad ezekre a kérdésekre.
A Beowulf szemléletével szemben az openMosix-telepben nincsen központi gép, mindegyik csomópont egyenrangú. A csomópontok (node) egy-, illetve többprocesszoros gépek is lehetnek, amelyeket helyi hálózaton (LAN) keresztül kell összekapcsolnunk. Egy egyszerű beállítófájlban, amelynek felépítését a későbbiekben részletezem, meg kell adni, hogy mely gépekkel osztja meg az erőforrásait, illetve mely csomópontok erőforrásait használja. Ha helyi hálózaton keresztül észleli a másik gép jelenlétét, erőforrásaikat rögtön megosztják. Ez az elgondolás nagyon rugalmassá teszi, mivel nincs rá szükség, hogy egy gép „közösködjön" a hálózatba kapcsolt összes géppel. Természetesen lehetőség nyílik arra is, hogy bármelyik gép erőforrásainak megosztását visszavonjuk. Ekkor egyszerűen visszahívja a távoli gépen futó folyamatokat és a saját processzorán futtatja őket tovább.

```
# /etc/openmosix.map

1     10.0.1.1      1
2     10.0.1.254    1
2     10.0.2.254    ALIAS
3     10.0.2.1      1
```

Tapasztalatok

Ahogy a hivatalos weblapon is olvashatjuk: „MOSIX!= openMosix". Nem véletlenül, a két elnevezés gyakran keveredik. Megesik, hogy a letöltött openMosix folttal (patch) kezelt rendszermag beállítása közben csak Mosixot emleget vagy a menü neve *openMosix*, de ezen a szinten csak Mosix elnevezésű beállításokat találunk és fordítva. Kipróbáltam a MOSIX-1.5.7-es változatát is, és gyakran előfordult, hogy a másik gépen a folyamatok ragadtak, amin csak egy újraindítás segített. Mindenesetre a Mosix- és openMosix-magokkal szerelt gépeket nem lehet vegyesen használni. Én a Debian GNU/Linux Woody változatára telepítettem (`kernel 2.4.17-openmosix`) minden csomóponton, és tökéletesen tette a dolgát. Háromgépes telepet hoztam létre (két 166 MMX, egy 533Celeron) 100 Mbit/s UTP-kábeles ethernethálózaton, ebből is látni, hogy nem kell erőmű a kipróbálásához. Amin Linux-rendszermag futhat, azt biztosan fürtbe tudjuk kötni. Kipróbáltam többprocesszoros géppel is (Dual 533 Celeron), ezzel sem volt gond. Öröm volt nézni, ahogy a `mosmon` parancsot (illetve `mon-openmosix`) futtatva a gépek terheltsége rövid időn belül közel azonos szintet ért el. Egyik 166 MMX gépen egy videokódoló program indítása után rögtön átkerült a másik gépre, és mindössze 100–200 KB/s nagyságú forgalmat hozott létre – százszázalékos processzorterhelés mellett. Ha nem futtatunk erőforrásigényes alkalmazásokat, akkor is 3–7 KB/s nagyságú forgalmat tapasztalhatunk. Most nézzük, hogyan érhetjük el mindezt!

Telepítés

Először is a rendszermagunkat kell megfelelő állapotba hoznunk. A ➲ http://openmosix.sourceforge.net/ címről tölthetjük le a legújabb rendszermagfoltot (kernel-patch) vagy Debian/GNU Linux alatt egyszerűen adjuk ki a következő parancsot:

```
apt-get install
➥kernel-patch-openmosix
```

Ez 2.4.16-os rendszermaghoz készült. Ha nem akarunk ennyire beavatkozni Linuxunk lelkivilágába, egy `user-mode-linux` rendszermagot is letölthetünk – képességeit saját főkönyvtárrendszerrel (rootfs) futtatva próbálhatjuk ki. Miután letöltöttük a rendszermagfoltot, másoljuk a rendszermag könyvtárába, majd a megfelelő könyvtárában rendszergazdaként adjuk ki a következő parancsokat:

```
zcat openMosix-2.4.17-2.gz | patch -p1
&& make menuconfig
&& make dep
&& make clean
&& make bzImage
&& make modules
&& make modules_install
```

A beállítások jelentése

- `openMosix process migration support`: bejelölésével a folyamatokat önműködően elosztja a csomópontok között, hogy a terhelésüket kiegyenlítse.
- `Support clusters with a complex network topology`: bejelölésével a csomópontok összetett hálózaton is összeköthetők.
- `Stricter security on openMOSIX ports`: biztonsági beállítás, hogy az olyan TCP/UDP-kapukat (port), amelyeket a Mosix használ, felhasználó ne érje el; hogy az erőforrásokat ne használhassa olyan csomópont, ami nincs a beállítófájlban felsorolva; hogy más folyamat ne használhasson olyan kapukat, amelyek a Mosixnak vannak fenntartva.
- `Level of process-identity disclosure (0-3)`: beállítja, hogy mennyi adatot tároljon az egyes folyamatokról.
  0 esetén: nincs további adat; 1: PID (/TGID, ha különbözik a PID-től);
  2: PID (/TGID), UID, GID; 3: PID (/TGID), UID, GID, PGRP, SESSION, COMMAND.
- `Create the kernel with a "-openmosix" extension`: a rendszermag egy *-openmosix* kiterjesztést kap.
- `openMOSIX File-System`: bejelölve openMosix-fájlrendszert (MFS) használhatunk. Erre akkor lehet szükségünk, ha egy csomópont fájlrendszerét az összes többi csomóponton el szeretnénk érni.
- `Poll/Select exceptions on pipes`: engedélyezi, hogy értesítsen egy programot, ha a csőből (pipe) olvasnak. Beállítva a `ioctl` (`pipefd`, `TCSBRK`, `arg`) is használható lesz, hogy kivételeket állíthassunk be, illetve törölhessünk

(exception conditions). Ha az (`arg & 1`) igaz, akkor – ha valaki olvasott a csőből – kivétel keletkezik. Ha az (`arg & 2`) igaz, akkor keletkezik kivétel, ha már nem olvasnak a csőből. Az alapbeállítás az, hogy egyik esetben sem keletkezik kivétel. Egy kivétel kezelője visszatérhet a `select` rendszerhívással, ami azt okozhatja, hogy a `poll` rendszerhívás visszatérési értéke belekerül a `POLLPRI` -be. Megkaphatjuk annak alsó becslését, hogy hány bájtot próbáltak olvasni a folyamatok a csőből az `ioctl` (`pipefd`, `TIOCGWINSZ`, `0`) segítségével.
Létezik olyan rendszermagfolt, ami lehetővé teszi a közvetlen fájlrendszerelérést (Direct File-System Access – DSFA). Csak akkor vehetjük igénybe, ha a Mosix-fájlrendszert is bekapcsoltuk.
Ha a rendszermag kész, állítsuk be az indításkezelőt (boot manager), és telepítsük az openMosix-csomagot (Debian: `apt-get install openmosix`).

Beállítás

Debian/GNU Linux alatt a */etc/openmosix.map*, Mosix esetén pedig a */etc/mosix/mosix.map*) fájlban kell megadnunk a hálózati csomópontokat. A beállítást elvégezhetjük kedvenc szövegszerkesztőnkkel vagy az `update-cluster` programmal is. Meg kell adni a csomópont azonosítószámát, IP-címét és a csomópontok számát a tartományon. Lehetőség nyílik külön alhálókon lévő gépek összekapcsolására is. Ekkor az átjáró mindkét (vagy több) IP-címét meg kell adni. Ha például két alhálónk (10.0.1.0/24, 10.0.2.0/24) és egy átjárónk (10.0.1.254, illetve 10.0.2.254 IP-címekkel) van, az *1. listában* szereplő módon kell beállítani.
Ha már az openMosixot támogató rendszermag fut, indítsuk újra az openMosixot: `/etc/init.d/openmosix restart`. Ha még nem az fut, a gépet az új rendszermaggal indítsuk újra. Mindezt az összes csomóponton meg kell tennünk.
A `mosid` fájlrendszer használatba vétele hasonlóan egyszerű. Először hozzuk létre az *mfs* könyvtárat: `mkdir /mfs`. A */etc/fstab* fájlhoz hozzá kell adnunk egy sort a *2. listán* látható módon. Ekkor a közvetlen fájlrendszerelérést (DFSA) bekapcsoltuk, ami az `mfs_mnt` a */mfs/[openMosix_ID]/* könyvtáron keresztül minden csomóponton elérhető lesz. Ha mindent jól csináltunk, már működik is, és elosztja az erőforrásokat.
Futtassunk nagy erőforrásigényű alkalmazásokat, és nézzük, hogyan nő a többi gép terhelése. Mindezt megtehetjük a `mosmon` parancs futtatásával, illetve attól függően, honnan szereztük be, `mon`, `mon -openmosix` néven találjuk meg. Ha elindítunk egy folyamatot, akár rögtön másik gépre költözhet (migrate), ami a telep (cluster) bármelyik gépe lehet. Ekkor a folyamat két részre szakad: egy felhasználói és egy rendszerszintű részre. Csak a felhasználói rész kerülhet át egy távoli csomópontra, míg a rendszerszintű mindig ott marad, ahol a folyamatot indítottuk. Az utóbbi felelős a rendszerhívások feloldásáért. Éppen ezért az olyan folyamatok, amelyekben gyakoriak a rendszerhívások, csak kis részben költöztethetők át másik csomópontra, vagyis kismértékű gyorsulást tapasztalhatunk. Ha tudni akarjuk, hogy valamelyik folyamatunk hol fut, futtassuk ezen a csomóponton a következő parancsot: `cat /proc/<pid>/where`. A parancs egy azonosítót ad vissza, ami annak a csomópontnak a sorszáma, ahol a folyamat fut. Mindezt egyszerűbben is megtudhatjuk, ha feltelepítjük az `mps` csomagot (Debian: `apt-get install mps`). Az `mps` a `ps` parancshoz, az `mtop` pedig a `top` parancshoz hasonlóan működik, ráadásul hasznos adatokat is megjelenít (melyik csomóponton fut a folyamat stb.). További hasznos tájékoztatást találhatunk a */proc/mosix* könyvtárban (mely csomópontokat éri el, mekkora azok terheltsége, milyen folyamatokat használ stb.).

```
# /etc/fstab

# tobbi fs beallitasai
# ...

#   [device_name]                          [mount_point]
mfs      defaults    0 0    mfs_mnt       /mfs
➥mfs    defaults,dfsa=1      0 0
```

Kapcsolódó címek

- http://www.openmosix.org
- http://www.mosix.org
- http://howto.ipng.be/Mosix-HOWTO
- http://openMosix.sourceforge.net
- http://packages.debian.org/unstable/net/openmosix.html
- http://packages.debian.org/unstable/net/kernel-patch-openmosix.html
- http://packages.debian.org/unstable/net/mps.html

Összefoglalás

Egyszerű használata és rugalmassága következtében hatékonyan alkalmazható üzleti célokra, otthoni gépek összekapcsolására, illetve egyetemi vagy középiskolai gépparkok teljesítményének, kihasználtságának növelésére. Napjainkban az egyetemeken egyre inkább elterjednek a linuxos telepek, amelyek olyan teljesítményt nyújtanak a kutatások számára, ami mindeddig nem volt elérhető. Felhasználási területeinek száma korlátlan, akár Beowulf-telepeket is összeköthetünk vele.

Kolcza Péter
(kpeter@sysconfig.hu)

A hónap szakmai tanácsai

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

Magas számú DHCP-kódokat kell beállítanom

Mivel az IP-telefónia területén dolgozom (Nortel), DHCP-kiszolgálómban a szolgáltatóra jellemző kódokat kell használnom. Létezik olyan DHCP-kiszolgáló Linux alá, amelyben be lehet állítani a Red Hat-terjesztésben használtaknál magasabb szolgáltatókódokat?
Bjoern Arstad, chancho@online.no

Ha a DHCP-kódodnak nincs neve, a `dhcpd.conf`-ban adhatod meg a következő módon:

```
option option-nnn 'value·;
```

Az `nnn` egy három számjegyből álló decimális kód. Olvasd el a `dhcp-options-dhcpd` súgóoldalt is.
Don Marti, dmart@ssc.com

A friss Red Hat-telepítés nem indul el

Red Hat 7.2-t próbáltam telepíteni egy Intel számítógépre, amelyben egy DPT VI (Adaptec) RAID 5 lemezvezérlő kártya volt. A telepítés rendben zajlott és létrejöttek a lemezrészek. A telepítés befejeztével a gép újraindításához megnyomtam az ENTER-t, de rendszerindulás helyett az alábbi hibaüzeneteket kaptam:

```
creating root device
mounting root filesystem
mount: error 19 mounting ext3
pivotroot: pivot_root (/sysroot,
➥/sysroot/initrd)      failed:2
freeing unused kernel memory: 220K
➥freed
kernel panic: No init found  Try
➥passing init=option to kernel
```

Byron Rendar, brendar@pcc.edu

Ted Ts'o megoldotta ezt a gondot; írása a ➲ http://www.redhat.com/mailing-lists/ext3-users/msg03575.html címen olvasható. Módosítanod kell a */etc/fstab*-ot; ha a fő fájlrendszert nem sikerül `ext3`-ként befűzni, próbáld `ext2`-ként. A */etc/fstab*-ban az `ext3`-bejegyzést változtasd át `ext2,ext3`-ra (később átalakíthatod az `ext2`-t `ext3`-ra). Egy másik, hosszabb megoldás szintén elérhető a Weben.
Chad Robinson, crobinson@rfgonline.com
Don Marti, dmarti@ssc.com

Internetezés kábelmodemmel

Hogyan kell az Internetre digitális kábelen keresztül csatlakozni? Szolgáltatóm a Cox Oklahoma Cityben, de ők nem támogatják a linuxos gépeket. Hálózati kártyám egy Realtek 8139 10/100. A SuSE Linux felismeri a kártyát és engedélyezett rajta a DHCP. A rendszerindítás során a Linux felismeri a kártyát, de IP-címet nem kap.
Matt Reynolds, mattreynolds@cox.net

Biztos, hogy a szolgáltatód DHCP-protokollt használ? Sok szolgáltató áttért a PPPoE-protokollra, ami nem ugyanaz. A biztonság kedvéért kérdezd meg a helyi Cox-irodát, mert a Cox mindkét módszert használja – attól függ, hol laksz. A SuSE támogatja PPPoE-protokollt, a leírásban az ADSL vagy T-DSL kulcsszavaknál szerepel a téma. A megfelelő YaST2 beállítóképernyő a `DSL configuration`. A `pppoed` csomagot telepítsd, és olvasd el a *sdb.suse.de/sdb/en/html/hoe_adsl_pppoe.html* oldalt.
Chad Robinson, crobinson@rfgonline.com
Don Marti, dmarti@ssc.com

Érintőképernyő beállítása

Mit kell beállítani az *XF86Config-4* beállítófájlban, hogy használni tudjam a Dynapro (3M) érintőképernyőt? Külön kell engedélyeznem az X indítása után az `xsetpointer NFI3` parancshoz hasonlóval?
Shane Kennedy, skenn@indigo.ie

A 3M (korábban Dynapro) érintőképernyők frissített meghajtóit a ➲ http://www.cdp1802.org/mmmtouch címen találod.
Robert Connoy, rconnoy@penguincomputing.com

A Perl nem nyomtat új sor nélkül

Nem tudok nyomtatni! Például `print 5;` kiadása esetén a szám nem íródik ki a képernyőre, csak ha egy új sor karaktert is mögé írok: `print 5,'\n';`. Ha a parancsértelmezőt `csh`-ra vagy `ksh`-ra változtatom, a hiba megszűnik. Bash vagy `sh` alatt a Perl-parancsfájl nem írja ki a számot a képernyőre, a karakterláncokat azonban igen. Az a kérdésem, mit kellene megváltoztatnom a Bash-parancsértelmezőben, hogy a Perllel helyesen működjön?
Blake Brezeale, blake.brezeale@bigfoot.com

Valószínűleg elrontottad a Bash-környezetedet. A parancssorban add ki a `reset` parancsot, és nyomd meg az ENTER-t.
Usman S. Ansari, uansari@yahoo.com

MAC-cím megváltoztatása

Szeretném megváltoztatni ethernetkártyám MAC-azonosítóját nagyjából úgy, ahogyan a Linksys útválasztóval ezt megtehetem. Létezik valamilyen egyszerű módja?
Mike O'Doherty, mgi1356@motorola.com

Nem írtad, hogy milyen ethernetkártyád van. Némelyikkel ez megtehető, míg másokkal nem. Ha a kártyád támogatja, erre a célra az `ifconfig`-ot használhatod:

```
ifconfig eth0 hw ether 001122334455
```

A DSL- és kábelmodemek felhasználóinak érdemes megjegyezniük ezt a trükköt, ha a szolgáltatást windowsos gépre fizették elő, és linuxos tűzfalat, illetve átjárót szeretnének. A legtöbb szolgáltató a MAC-cím alapján azonosítja a számítógépet, és ha az megváltozik, bonyolult feladat lesz átíratni a szolgáltatóval a változást. A fenti paranccsal a Linux-átjáró ugyanazt a MAC-címet mutatja kifelé, mint az eredeti ügyfélrendszer.
Chad Robinson, crobinson@rfgonline.com

Új termékek

Storix Backup Administrator

Megjelent a Storix Backup Administrator (SBA) 4.0-s változata, amely a legtöbb terjesztéssel, x86- és PPC-rendszereken használható mentő- és helyreállító program. A grafikus felülettel rendelkező SBA egy számítógép, vagy akár egy egész hálózat mentését megoldja. Beállítható a mentés időzítése, a teljesítménykimutatás jelentése, a felülírás és megtartás szabályozása, a szalag csíkozása és más sajátosságok.
Az SBA az adatokat az eredetitől eltérő vason is képes visszaállítani; a rendszer pedig a fájlrendszer típusának megváltoztatásával, a programból megvalósított RAID-eszközök hozzáadásával és az LVM-lemezrészek kialakításával újra testreszabható. A program minden jelentős LVM- és RAID-megoldást támogat.
Adatok: Storix Software,
➲ http://www.storix.com

CodeTEST for Embedded Linux

Az Applied Microsystems a CodeTEST for Embedded Linux megjelenésével belépett a Linux-piacra. A CodeTEST for Embedded Linux olyan programpróbáló és -elemző eszköztár, amely a fejlesztők számára a következő munkák elvégzését teszi lehetővé: átfogó teljesítménypróbák, memóriapróbák, valós idejű és forráskódszintű végrehajtás-követés, a rendszermag és illesztőprogram kódjának követése az eszközök szintjén, illetve az alkalmazások kódlefedettsége beágyazott Linuxon.
A CodeTEST egyszerre több mint 128 000 valós idejű függvényvégrehajtást tud mérni, hibákat keres az algoritmusokban, a híváspárelosztásokban és a segédeljárásokban, és méri a függvények által fogyasztott processzoridőt. A CodeTEST különálló modulok teljes rendszereként vásárolható meg.
Adatok: Applied Microsystems,
e-mail: info@amc.com,
➲ http://www.amc.com

Arkeia Virtual Server

Az Arkeia Corporation bemutatta a Virtual Servert, amely helyi és távoli adatvédelmi szolgáltatásokat valósít meg – kifejezetten az internetszolgáltatók, a telekommunikációs szolgáltatók és a kábelszolgáltatók igényeihez alakítva. Kétféle biztonsági mentést támogat. Az egyik az ügyfeleknek a szolgáltatóknál tárolt kiszolgálóinak helyi mentését végzi el, a másik az ügyfeleknél lévő gépek távoli mentését oldja meg a szolgáltatónál elhelyezett központi mentőkiszolgálón. Az Arkeie Virtual Server támogatja az egyidejű webhelyelhelyezést, nagy adatmennyiség esetén az egy gépről történő többszálú adatátvitelt, az ügyfélnyalábolást, az adatátvitelt korlátozott sávszélességű csatornán keresztül, valamint a gépek közti könyvtármegosztást. Az adatok védelméről a kinevezett hozzáférési kapuk, a mentett adatok egyéni katalógusai, a jelszavas és a proxys hitelesítés, és a nem megosztott mentési adathordozók gondoskodnak.
Adatok: Arkeia Corporation,
e-mail: sales@arkeia.com,
➲ http:// www.arkeia.com

PCI-8213 és PCI-8214

A PCI-8213 és a PCI-8214 két- és négykapcsos 64-bites gyors ethernetkártya, amelyek elkerülő képességgel rendelkeznek, így alkalmasak a tűzfalakban és a hálózati forgalommérőkben való használatra, hálózatos játékok játszására, és a nagy rendelkezésre állást kívánó, illetve internetes alkalmazásokban. Az elkerülőképesség abban áll, hogy a belső továbbító rendszerösszeomlás esetén vagy kikapcsolt állapotban keresztbe köti az 1-es és a 2-es csatornát. A hibatűrő adatátvitelt úgy érik el, hogy bejövő és a kimenő hálózati forgalmat csatolják, akkor is, ha a program vagy a gép lefagy. Mindkét kártya 64-bites 33, illetve 66 MHz-es PCI-sínen működik, de 32-bites PCI bővítőhelyekben is használható. Mindkét ethernetkapu két LED-del rendelkezik, az egyik a kapcsolatot jelzi, a másik a 10/100 állapotot.
Adatok: Adlink Technology America, Inc.,
e-mail: usa@adlinktech.com,
➲ http://www.adlink.com

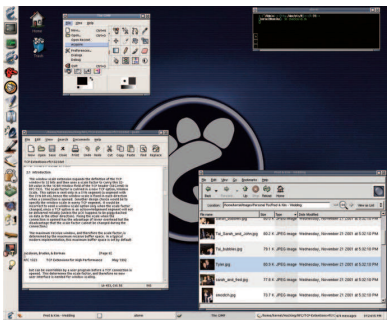
Rackable Systems 1U 1800

Xeon processzorokkal szerelik a Rackable Systems 1U 1800 kiszolgálóját. A Rackable 44U magas szekrényébe 88 darab 1U kiszolgáló fér, összesen 176 Xeon processzorral. A Rackable kiszolgálóiban a következő új Intel-termékeket használja még: SE7500CW2 kiszolgáló-alaplap, SE7500WV2 kiszolgáló-alaplap és SRSH4 kiszolgálófelület. Az új kiszolgáló a Rackable szabadalmaztatott számítógépházába szerelhető, amit úgy alakítottak ki, hogy a szabványos 19 hüvelyk széles és 30 hüvelyk mély szekrény első és hátsó felébe is, vagy a „Telco-féle” állvány mindkét oldalába beszerelhető legyen. Az 1800-as kiszolgálóba 8 GB memóriát, 1–4 merevlemezt, egy PCI X bővítőkaput, és egy vagy két 10/100/1000 ethernetkártyát szerelnek.

Adatok: Rackable Systems,
e-mail: sales@rackable.com,
➲ http://www.rackable.com

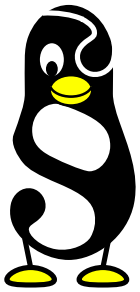
Gnome 2.0

A Gnome Foundation kiadta a Gnome munkaasztal és fejlesztői környezet 2.0-s változatát, amely a gyorsabb és többet tudó Nautilus fájlkezelővel, valamint új alkalmazások és segédprogramok tucatjaival rendelkezik. A 2.0-ban a beállítási folyamat is egyszerűsödött.



A kiadás tartalmazza a továbbfejlesztett GTK 2.0 eszköztárat, a `libxml2`-t, Glade-, Python- és CORBA-bővítményeket sok új programkönyvtárat és elemet. Az új változat tökéletesítései között szerepel a betűk élsimítása, a központosított és azonnal érvényre jutó rendszerbeállítás, az újraírt terminálalkalmazás fülekkel és profilokkal, a Yelp nevű súgóalkalmazás és még sok minden más.
Adatok: The GNOME Foundation,
➲ http://www.gnome.org

Linux Journal 2002. szeptember



A jogi tanácsadás megfelelő kerete egy jogász-ügyfél kapcsolat, amely egy adott helyzet minden tényállását figyelembe veszi, és a helyileg érvényes jognak felel meg. Bár ezt a cikket egy jogász írta, a benne foglalt adatok nem helyettesíthetik az esetre szabott, bejegyzett jogásztól származó tanácsadást.

A kockázatok megosztása

A terjesztési engedélyek arra szolgálnak, hogy a kibocsátó és az elfogadó fél között megosszák a kockázatot – az engedély jótállásra vonatkozó részének ez az egyik lényeges célja.

A BSD-engedélyben például a következő záradék szerepel: „Ezt a programot a szerzői jogok birtokosai és a közreműködők „úgy, ahogy van", mindenféle kifejezett vagy beleértett jótállás nélkül teszik elérhetővé, ami a minőségi megfelelőség és a meghatározott célra való alkalmasság biztosításának visszautasítását is magában foglalja, de nem korlátozódik csak rájuk. A szerzői jogok birtokosa vagy a közreműködők semmilyen körülmények között nem kötelezhetők semmilyen – közvetlen, közvetett, járulékos, egyedi vagy következményes – kár megtérítésére (ami magában foglalja a helyettesítő termék vagy szolgáltatás biztosítását, az elvesztett használati idő vagy adatok, az elmaradt nyereség vagy az üzletmenet folytonosságának megszakadásából adódó veszteségeket, de nem korlátozódik ezekre), függetlenül a kár keletkezésének módjától és a felelősség megállapításának hivatkozási alapjától, legyen az összefüggésben szerződésszegéssel, természetes felelősséggel vagy (gondatlanságra vagy más okra visszavezetett) sérelemmel, amely bármilyen módon kapcsolatba hozható a program használatával, még abban az esetben sem, ha a káreset bekövetkezésének lehetősége ismert volt.

Fontos észrevennünk, hogy a BSD-engedély feltételei mellett minden kockázatot az elfogadó fél visel. Mindegy, milyen borzasztó a program, mindegy, milyen kárt tesz a számítógépünkben vagy az üzletmenetünkben, mindegy, hogy a kibocsátó milyen ígéreteket tett a program hasznosságáról és képességeiről a reklámjaiban, a programot az engedély értelmében „úgy, ahogy van", mindenféle biztosíték nélkül fogadjuk el. Az összes többi nyílt forrású terjesztési engedély, amelyet olvastam, hasonló záradékot tartalmaz a jótállás és a kártérítési kötelezettség visszautasításáról. Az elfogadó fél visel minden kockázatot, amely a program használatából és az arra épülő programok létrehozásából származik.

Úgy gondolom, a kockázatok ilyenfajta megosztása igazságos, kivéve egyet, amelyet a BSD-engedély nem említ meg külön: annak kockázatát, hogy a kibocsátó fél a program terjesztési engedélyének kibocsátásához szükséges jogokkal eleve nem is rendelkezik. A BSD-engedély (a „magában foglalja… de nem korlátozódik ezekre" megfogalmazás miatt) kizárja a szerzői jogok betartására vonatkozó biztosítékot is.

A szerzői jogok betartásának biztosítéka nélkül abban az esetben, ha a kibocsátó valójában nem birtokosa az általa terjesztett program szerzői jogainak, illetve nem a szerzői jogok birtokosának engedélye szerint jár el, az elfogadó, nem pedig a kibocsátó fél viseli a szerzői jogi per kockázatát. Abban az Academic Free Licence nevű terjesztési engedélyben, amelyen jelenleg dolgozom, ezt a gondot a következőképpen orvosoltam:

„A kibocsátó szavatolja, hogy a program szerzői jogait birtokolja vagy a programot érvényes engedély hatálya alatt terjeszti. A megelőző mondatban külön megnevezett biztosítékot leszámítva a programot a szerzői jogok birtokosai és a közreműködők „úgy, ahogy van", mindenféle kifejezett vagy beleértett jótállás nélkül teszik elérhetővé, ami magában foglalja a minőségi megfelelőség, a meghatározott célra való alkalmasság, és a szerzői jogok megsértése elleni biztosítékok visszautasítását is, de nem korlátozódik ezekre. A kibocsátó, a szerzői jogok birtokosai vagy a közreműködők semmilyen körülmények között nem kötelezhetők bármiféle követelés, kárigény vagy egyéb kötelezettség megtérítésére szerződésszegési, kártérítési per útján vagy más módon, amely a programból ered vagy azzal kapcsolatba hozható."

A jótállási záradék első mondatában az engedély a szerzői jogsértés kockázatának terhét az elfogadó fél helyett a kibocsátó vállára helyezi. A elfogadó és a kibocsátó fél közül meggyőződésem szerint az utóbbi sokkal jobb helyzetben van annak megítélésében, hogy rendelkezik-e a szerzői jogokkal vagy olyan engedéllyel, amely lehetővé teszi számára a program terjesztési engedélyének mások számára történő továbbadását. Az elfogadó fél ezzel szemben nem rendelkezik azokkal az ismeretekkel, amelyek alapján eldönthetné, hogy elfogadja-e a szerzői jogok megsértésével járó kockázatot. Ezért az Academic Free Licence ezt a kockázatot arra a félre hárítja, amelyik a kockázat mértékét egyedül megítélni képes.

Ha például az engedély kibocsátója maga írta a programot, akkor tudja, hogy a szerzői jog is a birtokában van. Amennyiben csupán lemásolta valaki más programját, vagy egy cég alkalmazottjaként hozta létre a programot, nem szavatolhatja jóhiszeműen a szerzői jogok betartását. Sokan vonakodhatnak elfogadni a nyílt forráskódú programokat, ha az engedély kibocsátója nem ad valamiféle biztosítékot arra, hogy nem sérti meg a szerzői jogokat. Az ilyenfajta biztosíték hiánya csökkentheti a nyílt forráskódú programok elfogadhatóságát. Egy olyan szerzői jogokra vonatkozó biztosítékkal, mint amelyet az Academic Free Licence szövegébe foglaltam, az engedély elfogadói ésszerű mértékben hagyatkozhatnak arra, hogy az engedély megfogalmazása megvédi őket attól a vádtól, hogy nem foglalkoztak vele, ki a szerzői jogok birtokosa valójában.

Az Önök véleménye is érdekelne arról, hogy a nyílt forráskódú programok terjesztési engedélyeinek kell-e a szerzői jog megsértése elleni biztosítékot tartalmazniuk? Észrevételeiket az *lrosen@rosenlaw.com* címre küldjék – egy későbbi cikkben beszámolok e nem hivatalos felmérés eredményéről.

Linux Journal 2002. szeptember, 101. szám

Lawrence Rosen (➲ http://www.rosenlav.com) magángyakorlatot folytató jogász. A Nyílt Forrás Kezdeményezés ügyvezető igazgatója és jogtanácsosa (➲ http://ww.opensource.org).


Linuxvilág
www.linuxvilag.hu

A legeslegjobb linuxos gép

Találd ki, mi az: 1,1 Terabájt terjedelmű, 9,503 BogoMips-re képes, repülni tud és mindenki odavan érte?

Vajon mi különbözteti meg a „Legeslegjobb linuxos gépet” (Ultimate Linux box – ULB) a vadonatúj számítógéptől, amelyre Linuxot telepítettek? Otthoni rendszerünkhöz jól támogatott és megbízhatóan működő alkatrészeket válogatunk, és végül terabájtnyi adat tárolására képes eszközre teszünk szert egy szürke doboz formájában. Éppen az 1996 óta a Linux Journalban megjelenő „Legeslegjobb linuxos számítógépről” szóló cikkeken dolgoztunk. Abban az időben a Linux-kínálat kiterjedt az IBM nagyszámítógépekre, a 32-utas ccNUMA gépekre és más furcsaságokra is. Amennyire el szeretnénk ezek építési módját magyarázni, annyira nem rendelkezünk sem elhelyezésre alkalmas területtel, sem erőforrásokkal vagy akár költségvetéssel, hogy egy ilyen masinát kikapcsoljunk, valahányszor új hangkártyát szeretnénk beszerelni. Emiatt „Legeslegjobb linuxos gépünk” inkább amolyan „Legeslegjobb linuxos munkaállomás” vagy „Legeslegjobb linuxos kisméretű kiszolgálógép”, ami elég nagy ahhoz, hogy gyorsabb legyen, mint a legtöbb használati módban szükséges, és elég kicsi is ahhoz, hogy az otthoni vagy irodai környezetben is használható legyen.

A legelső kérdés természetesen az, hogy vajon meg kell-e a gépet vásárolni vagy jobb megépíteni? Egyrészről létezik az eszközök és a kívánságok listája – ezekről magad dönthetsz, ha saját kezűleg fejleszted a rendszeredet, ugyanakkor ezekkel a kérdésekkel a tömegpiaci számítógépgyártók nem nagyon foglalkoznak:

1. Válaszd a legjobb számítógépházat, amely a működtetni kívánt egységekkel szemben támasztott igényeidnek, a munkakörnyezetednek, a szépérzékednek és a belső szerelési igényeknek megfelel.
2. Használj csúcsminőségű tápegységet és berágódás ellen védett ventilátort.
3. Valóban jó SCSI-kártyát, SCSI-meghajtókat és ethernetkártyákat építhetsz be egy olyan gépbe, amely különben az asztali gépek csoportjának alsó kategóriájába (low-end) sorolható. Ez a fajta összeállítás jól fog működni linuxos munkaállomásként vagy kis csoportot kiszolgáló gépként, de a tömegpiacra termelő cégek rendszerint nem építenek ilyen gépeket, mivel ezek nem vonzóak az olyan emberek számára, akik a CPU-órajeleket és árakat mérlegre téve intézik beszerzéseiket.
4. Szabaduljunk meg az idővel porfogóvá váló alkatrészektől! A billentyűzetek és egerek a berendezések több nemzedékét képesek túlélni, sőt talán már CD-íróval felszerelt számítógép is fellelhető valahol.

Másrészről akad két terület, amelyről a tömegpiaci PC-gyártó képes gondoskodni, te egyedül viszont nem:

1. Csúcsminőségű hő- és zajszigetelés. Házi készítésű rendszeredben a nagyobb házba valószínűleg több ventilátor kerül, és a túlméretezett tápegység is nagyobb a hasonló adatokat felmutatni képes tömegpiaci társánál. De ha körültekintő vagy, mindennek nem kell szükségszerűen több zajt is jelentenie.
2. Kapcsolattartás az alkatrészgyártókkal, akik nincsenek rád tekintettel.

Jelenleg rengeteg nVidia-alapú kártya van forgalomban a piacon, és sok felhasználó nagy teljesítményről számolt be az nVidia-kártyákhoz tartozó saját meghajtó programok használatakor. Amennyiben a HP-nek dolgozol, szorosan együttműködsz az nVidiával, és titoktartási szerződés köt hozzá, akkor elképzelhető, hogy ez nálad is így van. A fogyasztók pontosan úgy viselkednek a HP-vel, mint bármelyik másik Unix-munkaállomásgyártóval: amennyiben baj van, hívják a HP-t, aztán majd ők elvégeztetik a hibakeresést az nVidiával, feltéve, ha ténylegesen az nVidia okolható érte.

Ha azonban központi programok új változataival dolgozol, amilyen mondjuk a rendszermag vagy az X-felület, akkor az egyes modulokra vonatkozóan nem sok segítséget várhatsz a hagyományos Linux-csatornákon keresztül. A USENIX-nél idén ***Linus Torvalds*** egyszer azt mondta: „Megesik, hogy működnek, azonban ilyen módon nem részesülsz a Linux összes előnyéből.”

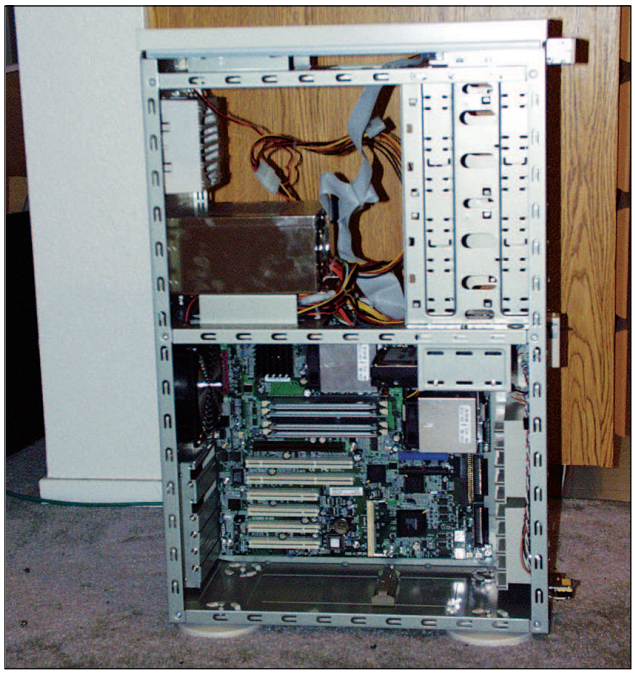
Az egyetlen eset, amikor egyáltalán elfogadhatnál egy nem linuxos rendszermagmodult vagy bármi hasonlót, akkor fordul elő, ha éppen a Linuxtól eltérő területtel foglalatoskodsz, vagy bármilyen más számítógéphez hasonlóan bánsz vele, és nem áll szándékodban rendszermagot fordítani. Ó, majdnem kifelejtettem még azt a lehetőséget, hogy esetleg teljesen megbízol a gép gyártójában!

A köztes megoldás a gép kis mennyiségben szállító kereskedőtől vagy a Linux-piacot kiszolgáló üzletből történő megvásárlása – érdemes ezt a lehetőséget számításba venni, ha kevesebb időt szeretnél vásárlással és a gép összeszerelésével tölteni, és mindemellett még Linux-barát alkatrészekre vonatkozó tanácsokat is kaphatsz. A linuxos alkatrészgyártók közvetlen és egyenrangú hálózatot alkotnak, és ez a rendszer meglepően jól működik Amerikában. Felkeresheted az élenjárók weboldalait, az összes, majdan a gépedbe kerülő alkatrész műszaki adatát átböngészheted, ráadásul a teljes rendszerhez feltétel nélküli jótállást kapsz.

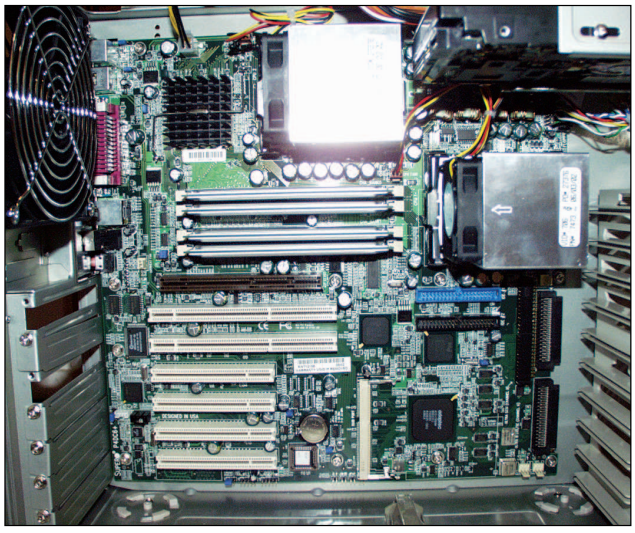
Idei „Legeslegjobb linuxos gépünk” kiindulópontjául az alapkiépítettségű, coloradói székhelyű Aspen Systems által szállított Glacier Dual Xeon gépet használtuk, és rögtön azt is elárulom, miért. Képzeljük el „Legeslegjobb linuxos gépünket” egy olyan Beowulf-elemként, amelyet toronyházba szereltek, és amibe jó minőségű grafikus gyorsítókártyát és hangkártyát építettek. A Beowulf-telepeket építő cégek a leggyorsabb központi egységek (CPU), alaplapok és tárelemek megbízható beszerzési forrásai, mert a kiszolgálógépek igen érzékenyek ezekre az alkatrészekre.

Központi egységek, alaplapok és tárelemek

Cikkünk sajnos az Intel cég RAMBUS-ról DDR-tártípusra való áttérése közben készült. Az írást még azelőtt nyomdába kellett adni, mielőtt a DDR- és AGP-támogatással ellátott Xeon-processzoros alaplapot a kezünkbe vehettük volna. Ez most már hamarosan meg fog jelenni. Ilyen előzmények után a RAMBUS-os SuperMicro P4DC6-os alaplapját kaptuk meg az Aspennél



1. kép
A nagy toronyházakban sok hely van, kényelmesen lehet velük dolgozni és a kábeleket is jól el lehet bennük rendezni. Az összes meghajtót a felső részben el tudjuk helyezni, ezért a levegőáramlás eszményi lesz. A képen kártyák és meghajtók nélküli ház látható.



2. kép
Olcsóbb megoldás ha a SCSI-vezérlő az alaplapon van, mintha egy külön kártyát vennénk. A nagy hűtők jók.

alkalmazásban álló ***Alan Taub*** javaslata alapján. Új gépünk 9503 BogoMips-re volt képes a 2.4.18-as rendszermaggal.
Minthogy cikkünket még a régi, RAMBUS súgtotta nyugtalanító időkben írtuk a DDR-világban élő boldog emberek számára, a legjobb, amit tehetünk, hogy egy csomó üres közhelyet puffogtatunk az alaplapi osztályon. Vedd a fáradságot és keresd fel a linuxos gépeket építő számítógép-forgalmazók weboldalait, és telefonon érdeklődj, hogy a felhasználói gondok mikor jelentkeznek náluk a leginkább.
Négy szolgáltatás a legtöbb alaplapon megtalálható, bár nem

3. kép
A SuperMicro a ház oldalára szerelt ventilátorokkal is segíti a hőmérséklet kordában tartását.

mindegyiken: SCSI-, ethernet-, hang- és grafikus csatolófelület. A kiszemelt alapok közül ne húzd ki valamelyiket csak azért, mert annak adott jellemzőjét éppen nem fogod kihasználni.
A linuxos hálózati kiszolgálópiac mérete miatt az összes ethernet-lapkakészletet – például az EtherExpress Pro 100-ast – meg fogod találni az alaplapokon, amelyeknek jó a támogatottsága. Ha ezzel szemben SCSI-egységekből álló rendszer építését tervezed, a SCSI-eszközöket támogató alaplap ára és az ilyeneket nem támogató alaplapok ára közötti különbség általában véve kevesebb, mint egy SCSI-kártya ára.
Az alaplapra szerelt grafikus gyorsító- és hangkezelő lapkák egyike sem tartozik a „Végleges linuxos gép” osztályba, de ha az alaplap kiszolgálógépként való későbbi újrafelhasználása is megfordult már a fejedben, ezek az alaplapok nem is kerülnek már olyan fájdalmasan sokba. Ha számos más irodai Linux-felhasználóhoz hasonlóan ritkán használod a hangkártyádat, talán nagyobb az esélye annak, hogy mégis használatba veszed, ha már egyszer az alaplapra szerelték.

Kép és hang

A Linux-rendszer építésének legkényesebb területe a háromdimenziós grafika. Idei választásunk egy ATI videokártyára esett szemben az nVidia kártyával. E döntésünk következményeiről ejt néhány szót keretes írásában ***Frank LaMonica***. A Monarch Computer Systems egy RADEON 8500 alapú Hercules 3D Prophet kártyával vett le bennünket a lábunkról, ezzel a tiszta képet adó fantasztikus 3D-s teljesítményű kártyával dolgoztunk.
A hangtámogatás még ennél is kellemesebb terület. Az ALSA-alapú rendszer megfelelő beállításával és ALSA alapú meghajtóprogramokkal nincs szükség az alaplapra épített hanglapka kikapcsolására ahhoz, hogy a csúcsteljesítményű hangkártyát üzembe helyezzünk: mindkettőt használhatjuk.
Az alaplapra épített eszköz ki- és bemenetét a konferenciarendszer számára kijelölt eszközként az egyes telephelyek közötti beszélgetésre használtuk, miközben a Sound Blaster Live! kártyát az Ogg Vorbis-állományok fejhallgatón történő lejátszására tartottuk fenn. A Sound Blaster Live! kártyát még mindig kedveljük, mindenütt jelenlevő rendszermagbeli támogatottságának, könnyű telepíthetőségének és jó hangminőségének köszönhetően. Ezenkívül fontos megemlíteni azt a tényt is, hogy az eszközt egyidejűleg 32 program használhatja.

Melyik 3D-támogatásos kártyát válasszam a Linuxhoz?

Töménetlen grafikával kapcsolatos fejlesztés zajlik, a rendszermagfejlesztéstől kezdve egészen a különböző alkalmazások által használt fejlett grafikus rétegekig. A rendszermag szintjén éppen a közelmúltban sikerült befejezni a Direct Rendering Manager (DRM) parancssori felületét. Az új belső DRM parancssori felület lehetővé teszi a meghajtóra jellemző rendszermag-kiterjesztések létrehozását, ugyanakkor karbantartja az eszközfüggetlen felületeket, amelyek az önállóan működőképes meghajtóprogramok támogatásához szükségesek.

A Linux-változatok szállítói másfajta DRM-munkát végeznek a DRM-támogatás érdekében, hiszen a grafikus szolgáltatások minél jobb támogatása mindegyikük számára fontos célkitűzés. Ezek rendszerint meglehetősen kis kezdeményezések, és nem is kapnak nagy nyilvánosságot.

A legérdekesebb csúcsszintű kezdeményezés a Chromium nevet kapta. A Chromium-kezdeményezés „rugalmas keret a méretezhető munkaállomásokból álló telepeken végzett valós idejű leképezés (rendering) számára, amely Stanford WireGL Project forráskód alapjára épül."

A Chromium a munkaállomásokat egyetlen nagy teljesítményű leképezőtelepként használja. Modulrendszerű felépítése lehetővé teszi a tetszőleges kirajzoló adatcsatornák kiépítését. A példák között találhatunk nagyméretű, több képernyőből álló falakat és párhuzamosan működő képfeldolgozó hálózatokat. A Chromium OpenGL API-t (alkalmazás-programfelületet) használ, ezért együtt használható a legtöbb háromdimenziós alkalmazással. Jellemző felhasználási területei: nagy mennyiségű adat tudományos szintű megjelenítése, többképernyős szórakoztatóipari képmegjelenítés, illetve felvételkészítés, és kutatási területként: párhuzamos kirajzolás.

A Mesa, az OpenGL nyílt forrású megvalósítása, amely minden DRI-meghajtó lelke, továbbra is figyelemmel kíséri a legfrissebb OpenGL-fejlesztéseket. A Mesa teljes mértékben megvalósítja az OpenGL 1.4-es szabványt, és a legfrissebb OpenGL Architectural Review Board (ARB)-kiterjesztéseket. *Brian Paul* a kezdeményezés alapítója, négy új ARB-kiterjesztésért volt felelős, és maga is hozzájárult az OpenGL 1.4-es meghatározás kialakításához. Mindegyik kiterjesztést tavaly hagyták jóvá.

Gépi szinten szigorúan őrzik a legújabb lapkatechnológiát. Az nVidia saját grafikus adatcsatornáival megírt sikertörténete más gyártók számára is kijelölte az utat: kövessék a példát, csak végrehajtható programokkal. A legtöbb lapkagyártó most állhatatosan megtagadja a termékek műszaki leírásának kiadását a Nyílt Forrás Közössége számára. Minthogy sokan támogatnak Linux-rendszeren futó zárt forrású meghajtóprogramokat, egyre nehezebbé válik meggyőzni a háromdimenziós lapkák gyártóit, hogy a nyílt forráskód alapelvének támogatása éppen az ő jól felfogott érdekük.

Jelen írásunktól kezdődően az ATI és Intel cég továbbra is a nyílt forráskód fejlesztői rendelkezésére bocsátja a leírásokat, úgy tűnik azonban, hogy csupán az Intel tart ki a tisztán nyílt forrású kezdeményezés mellett.

A saját fejlesztésű adatok visszatartása különböző okokhoz köthető, azonban azt már nagyon nehéz elvitatni, hogy a lapkagyártóknak olcsóbb, ha műszaki titkaikat megtartják, minthogy a szabadalmi keresések után kötelesek lennének fizetni, amiből megtudnák, hogy az éppen újonnan bejelenteni kívánt szabadalmuk nem tartozik-e máris valaki máshoz.

A peres eljárások sokkalta drágábbak, mint bármilyen közvetlen pénzügyi nyereség, amelyet a nyílt módszer hoz a gyártónak. Jelenleg a jogi hátteret az újítások akadályozására használják, és még maga az Egyesült Államok kormánya is erőtlen a szabadságjogok elvesztésének és rövid távú előnyökre váltásának a megakadályozásában.

Mindenképpen elismerést érdemel viszont, hogy a Weather Channel a Tungsten Graphics céggel aláírt egy szerződést, hogy egy ATI Radeon 8500-as kártya nyílt forrású meghajtóprogramjának fejlesztését fizeti. A programból hiányoznak bizonyos 8500-as működésjavító szolgáltatások, amelyeket az ATI kizárólag a bináris változatba fog beépíteni, csakhogy még azok nélkül is fontos eredmény a nyílt forráskód szempontjából.

Frank LaMonica

De hogy kerül a terabájtnyi adat az asztalra?

Mostanáig biztonságos és nagy teljesítményű, ám egyúttal drága tárolóeszköz választása mellett történik lándzsát: a nagy fordulatszámú SCSI-merevlemezek közül kettőt is találhatsz – ez még mindig a jó kis hagyományos lehetőség.

Amikor azonban a Szilicium-völgyben megrendezett Linux-felhasználói csoport egyik összejövetelén a 3ware munkatársai egy Escalade 7850-es Storage Switch eszközzel tűntek fel, elhatároztuk, hogy ezt is kipróbáljuk. Ez volt az első alkalom, hogy egy munkaállomásba RAID 5-ös rendszert és terabájtnyi (1024 gigabájt) adat tárolására képes eszközt telepítettünk. Munkára bírni sem nehéz.

A 7850-es típusjelű mikroprogramozott eszköz frissítéséhez a gépet MS-DOS indítólemezről kell újraindítani, viszont az eszközt támogató modul már szerepel a 2.4.18-as rendszermagban.

A dobozból kivett és beszerelt lemeztömböt két Linux-változat is képes volt felismerni. Meglepetéssel fogjuk tapasztalni, hogy a 3ware meghajtóprogramját a */usr/src/linux/drivers/scsi* könyvtárban fogjuk megtalálni, nem pedig a */drivers* könyvtár */ide* alkönyvtárában.

A rendszermag szempontjából az ATA-felületű RAID-meghajtó SCSI-felületű eszközként látszik. A 2.4.18-ast megelőző rendszermagokkal nem lehetett terabájt nagyságrendű állományrendszereket felépíteni, ezért győződj meg róla, hogy a rendszermagod 2.4.18-as vagy frissebb legyen.

A készülékhez adott webalapú eszközkezelő segédprogram – a neve 3DM – a tulajdonjog hatálya alá esik, a 3ware ugyanakkor azt állítja, hogy valamikor mostanában a GPL szerződésnek megfelelő, héjprogrammal irányítható parancssori eszközt fog megjelentetni.

A 3DM egyértelmű és könnyen használható felületet biztosít a program telepítéséhez és használatához. A webfelületű űrlapról a teljes RAID-tömb beállítása elvégezhető.

Ha a 3DM programot használatba szeretnéd venni, lehetséges, hogy a webböngésző programot újra be kell állítani. A szóban forgó program a 1080-as kaput használja, ezért a Mozilla ***A megadott kapuhoz történő hozzáférés biztonsági okok miatt nincs engedélyezve*** üzenetet fogja megjeleníteni. A hiba elhárításához az alábbi sort kell beszúrni:

```
pref("network.security.ports.banned.ovveride",
➥"1080");
```

a Mozilla ***all.js*** állományába, ami ***/usr/lib/mozilla/defaults/pref/*** könyvtárban található.
A 3ware állítása szerint a készülék képes a lemezhiba kezelésére fenntartott menet közbeni csere (hot swap) kezelésére, amennyiben megfelelő szerelőkeretben helyezkedik el. A magunk részéről azonban csak a meghajtóegységek szokásos telepítését végeztük el, és eltekintettünk az utóbb említett szolgáltatás kipróbálásától. Mindamellett remek működési adatokat kaptunk. Az ext3-állományrendszeren nyolc ATA-felületű Maxtor merevlemezzel 173,1 MB/s-os olvasási és 23,5 MB/s-os írási sebességet kaptunk eredményül az egyébként terheletlen rendszeren. Ez nagyjából 10 000 fordulat/perces SCSI-meghajtó teljesítményének felel meg, de az írási műveleteknél annál hatszor gyorsabb.
Az eszköz sebességének növelése érdekében ***Adam Radford***, a meghajtó programjának szerzője két ***/proc*** könyvtárbeli módosítást javasolt, amelyeket pontosan követtünk. Az előreolvasási jellemző (***/proc/sys/vm/max-readahead***) értékét 256-ra, míg a legkisebb értéket jelző ***/proc/sys/vm/min-readahead*** jellemzőt 128-ra módosítottuk.

A számítógépházakkal és a szereléssel kapcsolatos tanácsok

Bizonyára senki sem óhajtja megvalósítani a „Legeslegjobb linuxos számítógépről” alkotott elképzelést, amikor pusztán az első saját gépét szeretné összeszerelni, már csak az alkatrészárak miatt sem. Mentsük meg egy hibás merevlemezzel bíró asztali gép jó állapotban lévő dobozát, és egy meghibásodott alaplappal rendelkező kiszolgálógépet, vagy valami hasonlót.
A számítógép saját kezű összeszerelésekor a nagy toronyházak főnyereményszámba mennek: rugalmasságot biztosítanak, tetszés szerint bármit oda szerelhetsz be, ahová te szeretnéd. Továbbra is kedveljük a Lian-Li alumíniumházakat, amelyek egyikét már tavaly is használtuk.
A legkevésbé a számítógépházakra annyira jellemző piszkosszürkét szeretjük. Talán az angol nyelvterületen működő számítógépgyártók nem is véletlenül adták ennek a csúf színnek a Ragacs becenevet, hiszen az enyvre hajaz. Óriási, majd pont egy ilyen enyves dobozt szeretnék magam mellett tudni az odúmban!
Sajnos a más színű, szebb küllemű házak többségét főként játékkedvelőknek tervezték, akik forrón szeretik a központi egységeket és grafikus kártyákat, de ATA-felületű merevlemezekből csak egyet-kettőt használnak. Így bizony nehéz szívvel, de a SuperMicro sc760-as házsorozatot kell ajánlanunk, nem kizárólag nagy befogadóképességéért, de a merevlemez és ventilátor megfelelő elhelyezése következtében is.
A SuperMicro-házakban rengeteg kényelmes szerelési hely várja a merevlemezeket és ventilátorokat, amelyek közül sokat fel sem kell majd használnod. Az egyes változatok a Xeon-processzorok szereléséhez szükséges szerelőlyukakkal ellátott alaplapokat is támogatják, mások viszont nem, emiatt először az alaplapot kell kiválasztani, még mielőtt döntenénk a ház felől.
A házon belüli szerelés könnyű: távolítsd el az elülső zárólapot, ezután az oldallapok, mint valamiféle könyv lapjai, lenyithatók. A házat egy alaposan átgondolt toronyként képzeld el, vagy mint egy kétszintes épületet: az alaplap és a bővítőkártyák a földszinten helyezkednek el, a merevlemezek és a tápegység pedig az emeleten. Egy 12 cm átmérőjű kifúvó ventilátor található közvetlenül a földszinten lakó központi egység mögött. Az elülső oldalra viszont akár három beszívó ventilátort is szerelhetünk. Az emeleten egy kifúvó ventilátor helyezkedik el mindjárt a tápegység fölött, s a merevlemezek oldalaira további négy ventilátort lehet szerelni.
Azt javasoljuk, hogy a földszinti merevlemez-befogadó helyeket hagyjuk üresen, feltéve, hogy megtehetjük; és a merevlemezt a szerelőkeretben helyezzük az 5,25 hüvelykes helyre. Ez a megoldás több levegőbeszíváshoz használható területet hagy a földszinten, ugyanakkor a merevlemezeket a ventilátorok útjában tartja.
A 3ware cég saját szalagkábele a merevlemezekhez tartozó szalagkábelekkel együtt mintegy 27 cm²) tesz ki. A hanyagul vezetett tömérdek kábel képes a szabad légáramlást megakadályozni. A kábeldzsungelt lapos köteggé formáltuk és Velcro-szalaggal átkötöttünk.
Fontos a be- és kifúvó ventilátorok egyensúlya. A józan ész azt sugallja, hogy „a forró levegőt fújjuk ki”, a túl sok ventilátor azonban csökkenti a házban levő légnyomást, megsemmisíti a tápegység ventilátorának erőfeszítéseit, és csapdába ejti a forró levegőt.
A tápegység működésével egy másik fontos dolog is együttjár. Minthogy a forró levegő felemelkedik, az alulra szerelt elülső oldali beszívó és hátoldali kifúvó ventilátorokkal nehéz elvéteni a helyes megoldást.
Az SC760-as ház minden felülete festhető, fúrható, sőt egyetlen kábel kihúzása nélkül eltávolítható. Ha tehát éppen selyemcukor pirosra festve és szellőző lyukkal együtt szeretnéd használni, a szürke dobozt csak be kell adni egy mázolóműhelybe, ahol úgy átfesthetik, hogy a benne levő lemezeket, kábeleket meg sem kell mozdítani.

Vegyes észrevételek

Bizonyára magad is tapasztalni fogod, hogy nem ejtettünk szót az olyan kiszolgálóegységekről, mint a billentyűzetek, az egerek és a képernyők. Ezek mind az egyéni ízlés körébe tartoznak, és a célnak tökéletesen megfelel, ha valaki egy használtra tud szert tenni. Vajon hány számítógépet tudsz összeszámolni, amelyik túlélte a kedvenc billentyűzetedet vagy egeredet? És mi a helyzet a CD-írókkal, a DVD-meghajtókkal és a szalagos egységekkel? Egy felmérésben azt olvastuk, hogy olvasóink 95 százaléka több géppel is rendelkezik. Ezért azt javasoljuk, hogy a gépek közül az egyiket adatmentő- vagy CD-író számítógépnek állítsd be.
A fentiek talán már elegendő útravalóul szolgálnak egy saját linuxos gép építésének megkezdéséhez. Most már a SuSE-Linux telepítőprogram halhatatlan szavaival élve azt mondhatjuk: jó szórakozást!

Linux Journal 2002. szeptember, 101. szám

Don Marti
a Linux Journal szakmai szerkesztője, olvasóink a dmarti@linuxjournal.com címen érhetik el.

Kapcsolódó címek

ATI-termékek
➲ http://www.ati.com/products/builtdesktoppc.html
A Chromium-kezdeményezés
➲ http://sourceforge.net/projects/chromium
Az Intel 845G lapkája
➲ http://www.intel.com/design/chipsets/845g

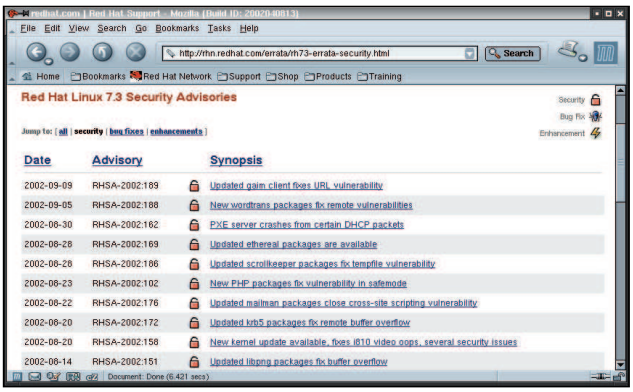
Biztonsági öveket becsatolni!

Amint rászabadulunk a Világhálóra, célpontjai lehetünk a kívülről érkező támadásoknak.

A Linux biztonsági szempontból alapvetően abban különbözik a Windows 95/98/ME rendszerektől, hogy senki sem teheti azt, ami éppen az eszébe jut. A rendszergazdát kivéve mindenkire komoly megkötések vonatkoznak. Nemcsak az, hogy az egyik felhasználó nem túrhat a másik felhasználó „cuccába”, de semmi olyasmit sem csinálhat, ami a rendszer biztonságát, illetve üzembiztosságát csak a legkisebb mértékben is veszélyezteti. A Windows esetében (kivéve az NT-ket és a 2000-et) semmi ehhez hasonló kijátszhatatlan védelemmel nem találkozhatunk. A Linuxban viszont ezek a védelmi módszerek mélyen az operációs rendszerben találhatók, tehát nincs út a kikerülésükre.

Biztonsági kiskáté

Legalábbis elméletileg. A Linuxot és alkalmazásait is emberek írták, tehát előfordulhatnak bennük hibák. Ezek a hibák (még a unixos időkből származó nevükön bugok) általában veszélytelenek, viszont nem egyszer volt rá példa, hogy ilyen programhibák kihasználásával olyan jogosultságokat lehetett szerezni a rendszerben, amilyen eredetileg nem járt volna nekünk.
Félreértés ne essék: a Linux biztonságos rendszer, de csak akkor, ha odafigyelünk bizonyos dolgokra. Hogy pontosan mire is, ez lesz sorozatunk jelenlegi részének témája.
Manapság az Internetről érkező legnagyobb fenyegetést a különböző vírusok, illetve trójai programok jelentik. A Windows CE (95/98/ME) rendszerekben az a gond, hogy a felhasználó bármiféle megkötés nélkül bármelyik állományt átírhatja, törölheti stb. A Linux- (és a Unix-) rendszerekben azonban az a bevált szokás, hogy a felhasználó egyetlen program binárisába sem piszkálhat bele. Sőt, írásjoga csak a saját könyvtárára van és az alkalmazásokhoz tartozó személyes beállításait is kizárólag itt tárolhatja.
Ez a módszer hatékony védelmet nyújt a vírusokkal szemben. Ha egy átlagos felhasználó (azaz nem rendszergazda) elindít egy vírust tartalmazó programot, az nem képes más alkalmazásokat megfertőzni, mivel a felhasználónak nincs írási jogosultsága, csak olvasni és végrehajtani tud. Ha azonban rendszergazdaként indítjuk el ugyanazt a fertőzött programot, kellemetlen helyzet állhat elő: a vírus tetszés szerint bármelyik másik alkalmazásba is befészkelheti magát.
Ezért a legfontosabb alapszabály, hogy sohase lépjünk be fölöslegesen rendszergazdaként, csak akkor, ha feltétlenül szükséges! Ha valamilyen rendszerbeállítást szeretnénk megváltoztatni vagy egy új alkalmazást telepíteni, azt természetesen csak rendszergazdaként tehetjük meg. De sohase böngésszünk rendszergazdai jogosultságokkal az Interneten, és így bejelentkezve ne indítgassunk el innen-onnan szerzett programokat! Ez az oka annak, hogy a legtöbb terjesztés már a telepítés folyamán létrehozat velünk egy saját felhasználót, amellyel majd dolgozni fogunk.
A másik fontos feladat jelszavunk helyes megválasztása és titokban tartása. A Linux- (és a többi Unix) rendszer távolról is felügyelhető, azaz nincs olyan dolog, amit csak a konzolról tudnánk megtenni. Ha gépünk futtat `telnet`, illetve `ssh`



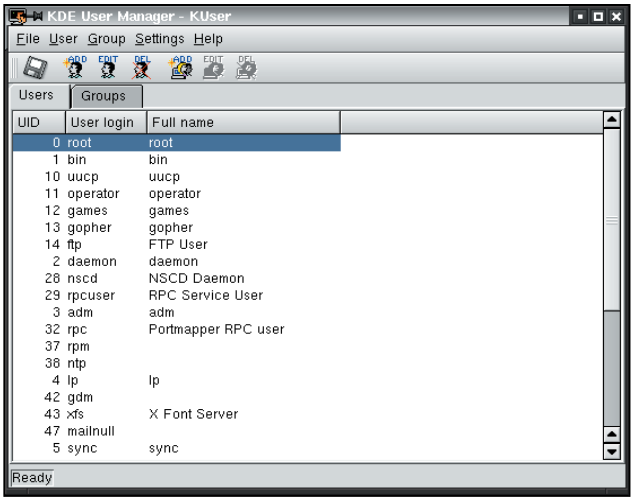
Legalább havonta egyszer keressük fel a Linux honlapját, és nézzük meg, milyen új hibákra derült fény az elmúlt időszakban

(a `telnet` titkosított „változata”) szolgáltatást, a megfelelő jelszavak ismeretében bárki megkaphatja Linuxunk parancssorát. Igaz, általában nincs megengedve, hogy távolról közvetlenül rendszergazdaként jelentkezzünk be. Először ezt a saját felhasználói nevünkön kell megtennünk, majd az `su` parancs kiadásával és a rendszergazdai jelszó begépelésével kaphatjuk meg a rendszergazdai parancssort.
Akinek jelszavaink a birtokába kerülnek, annak csak pillanatnyi IP-címünket kell kiderítenie, és már gyakorolhatja is az uralmat gépünk felett. Ezért ne használjunk könnyen kitalálható jelszót, sőt ha erős üldözési mániánk van, a `telnet`, illetve az `ssh` szolgáltatás elérését akár korlátozhatjuk is (ennek miként jét rövidesen részletesen is bemutatjuk).
Ha ezt a két irányelvet betartjuk, máris sokat tettünk rendszerünk biztonsága érdekében, de még korántsem mondhatjuk atombiztosnak.

Egy kis démonológia

A rendszerekbe ugyanis úgy is be lehet törni, hogy kihasználják bizonyos programok hibáit, illetve hiányosságait. Egy kívülről érkező támadó szerencsére csak a gépünk nyújtotta szolgáltatások biztonsági réseit próbálja kihasználni. Ha tehát egy FTP-kiszolgálót futtatunk, gépünk FTP-szolgáltatást nyújt, és az FTP-kiszolgálódémon számára kívülről adatokat lehet küldeni. Az FTP-démon általában rendszergazdai jogosultságokkal fut, és könnyen előfordulhat, hogy bizonyos programhibából eredően arra is rávehetjük, hogy úgynevezett idegen kódot (a támadó által írt programot) futtasson le. Ha az FTP-kiszolgáló rendszergazdai módban fut, az idegen kód is örökli ezeket a jogköröket, tehát a betörés sikeres volt.
A tanulság tehát az, hogy minden általunk futtatott szolgáltatás (például FTP, Web stb.) lehetséges behatolási pont a rendszerünkbe. Ezért a harmadik alapelv: csak olyan szolgáltatásokat futtassunk, amelyekre valóban szükségünk van.
Hogy milyen szolgáltatásokat futtatunk, azt a legegyszerűbben kapuink pásztázásával tudhatjuk meg. Linux alatt az egyik legnépszerűbb kapupásztázó program az `nmap` (lásd Linux-

világ 6. szám, 45–49. oldal), amit a legtöbb Linux-változat is tartalmaz. Használata rendkívül egyszerű, csak írjuk be az `nmap localhost` utasítást, és rögvest kiderül, hogy éppen milyen szolgáltatások futnak a gépünkön.
Az `nmap` nemcsak a kapu számát, hanem a futó szolgáltatás nevét is kiírja nekünk. Csak azokkal a kapukkal érdemes foglalkoznunk, amelyek `open` állapotúak. Sorozatunk egyik korábbi részében már említettük, hogy a szolgáltatások megvalósítása



Sose használjuk a rendszergazdai jogosultságot az Interneten való barangolásra. Ha idáig nem tettük volna meg, hozzunk létre egy külön – korlátozottabb jogkörökkel bíró – felhasználót

a démonok feladata. Minden démon más-más szolgáltatásért felelős. A feladatunk annyi, hogy a megfelelő démont leállítsuk, és ezután az általa nyújtott szolgáltatás már nem lesz elérhető (a démonokat, leállításukat és újbóli elindításukat sorozatunk negyedik részében már áttekintettük, lásd Linuxvilág 16. szám, 38–39. oldal). Sajnos nem mindig egyszerű kitalálni, hogy az adott szolgáltatás melyik démonhoz tartozik, de segítségünkre lehet a folyamatlista (`ps ax` parancs), ebben tekinthetjük meg a futó démonok (és egyéb alkalmazások) névsorát.
A démonok tárgyalásakor szót ejtettünk az `inetd`-ről (Internet Daemon) is. Ennek célja, hogy ne kelljen folyamatosan futtatnunk azokat a démonokat, amelyekre viszonylag ritkán van szükségünk. Mint tudjuk, a démonok csak akkor képesek dolgozni, ha munkájuk is van, egyébként a háttérben unatkoznak, viszont a memóriát ugyanúgy foglalják. Ha például olyan kiszolgálót üzemeltetünk, ahol az FTP-szolgáltatást csak weblapunk frissítésére használjuk, az FTP-démont gazdaságtalan folyamatosan futtatni, mivel átlagosan napi 1–2 alkalommal lehet rá szükségünk. Ugyanakkor a webkiszolgálót nem érdemes ki-bekapcsolgatni, mert egyrészt viszonylag gyakran szükség van rá (weboldalunkat naponta akár több százan, esetleg ezren is megnézhetik), másrészt indítása és leállítása hosszadalmas feladat, tehát lelassulna a kiszolgálás sebessége.
A megoldás: a webkiszolgálót igen, de az FTP-démont ne indítsuk el a rendszer betöltése után, hanem bízzuk az `inetd`-re! Az `inetd` észleli, ha egy kapun kívülről kapcsolatot akarnak kezdeményezni. Ekkor megnézi, hogy az adott kapuhoz melyik szolgáltatás van hozzárendelve, majd elindítja a megfelelő démont. A démon a kiszolgálást követően nem marad a memóriában, hanem befejezi a futását.
Egy otthoni rendszeren érdemes minél több szolgáltatást az `inetd`-be pakolni. Számos terjesztés esetén seregnyi felesleges szolgáltatás be van állítva, amely nem feltétlenül jelent veszélyt a rendszerre, de nincs is sok értelme benne hagyni. Az `inetd` beállításait az */etc/inetd.conf* állományban találhatjuk.
A fájlban minden sor egy-egy szolgáltatást jelöl. Az első oszlop a szolgáltatás neve. A második, harmadik és negyedik oszlop a kapcsolat típusára vonatkozik. Ezután a felhasználó neve következik, akinek jogosultságaival az utolsó oszlopban megadott démon futhat. Ez általában a rendszergazda.
Ha egy szolgáltatást ki szeretnénk iktatni az `inetd`-ből, egyszerűen tegyünk egy # (kettős keresztet) a megfelelő sor elejére. Ne felejtsük el, hogy a beállítás csak akkor lép érvénybe, ha előtte újraindítjuk az `inetd`-t.
Sok terjesztés (mint például a Red Hat is) az `inetd` helyett másikat használ (például az `xinetd`-t – eXtended Internet service Daemon).
Minden szolgáltatáshoz külön beállítófájl tartozik, amelyeket a */etc/xinet.d/* könyvtár alatt találunk. Amennyiben egy szolgáltatást ki szeretnénk iktatni, a `disable = no` sort a megfelelő állományban állítsuk `disable = yes`-re.
Ha ezzel készen vagyunk, keressük fel Linux-változatunk honlapját, és nyálazzuk végig a hibalistát. Leginkább azokra a hibákra összpontosítsunk, ahol fel van tüntetve, hogy a rendszer biztonságát is veszélyezteti. Ebben az esetben azt is megtudhatjuk, hogy az adott biztonsági lyukat csak belülről (local) vagy kívülről is (remote) is ki lehet-e használni.
Mit jelent ez? A támadásokat két csoportra bonthatjuk: belsőre és külsőre. Külső támadás esetén a támadó nem rendelkezik számlával (account) a rendszerben, azaz nem férhet hozzá a parancssorhoz, tehát például nem futtathat különböző programokat. Kizárólag csak a hálózati szolgáltatásokat nyújtó démonokat „zaklathatja”. Belső támadásról akkor beszélünk, amikor sötét lelkű hősünk bejelentkezik vagy `telnet`-en, vagy `ssh`-n keresztül és a rendszergazdai jogkörökkel futó programok biztonsági réseit próbálja kihasználni. Ilyenkor tehát jóval nagyobb a támadási felület, viszont a betörőnek ismernie kell egy felhasználói nevet és a hozzá tartozó jelszót.
Egy otthoni rendszer esetében nyilvánvalóan csak a külső támadásokra érzékeny hibákat kell befoltoznunk (kivéve, ha fűnek-fának elérést adunk), de azt javasoljuk, hogy egyetlen ismert biztonsági rést se hagyjunk nyitva. Különben is egy hibás program „befoltozása” nem ördöngös művelet: Linux-változatunk honlapján le van írva, hogy egyrészt merre találjuk a frissítést, másrészt mit kell vele tennünk. Egy RPM-csomag esetében általában elég, ha kiadjuk az `rpm -Fvh csomag.rpm` parancsot.
Ha ez is megvan, nem árt végiggondolnunk: valóban szükséges-e, hogy egy adott szolgáltatás a világ bármely pontjáról elérhető legyen? Például bizonyos címtartományokra korlátozhatnánk, amire több út is kínálkozik, ám ez már a következő rész témája, amelyben alapszinten megismerkedünk a Linux-rendszermagban lévő úgynevezett állapotfüggő csomagszűrő tűzfallal.
Ez egy otthoni rendszeren is hasznos lehet, például megvédhet bennünket a DoS- (Denial of Service) támadásoktól, de ugyanitt állíthatjuk be a több gép közti internetmegosztást is (masqueradingbújtatás). A részletekről egy hónap múlva ugyanitt.

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.

Ha bemegyek...

A folyamatkezelés mellett a beviteli-kiviteli eszközök (például billentyűzet, monitor, merevlemez, nyomtató) kezelése is fontos része az operációs rendszernek. E feladat megvalósítása az úgynevezett beviteli-kiviteli programra hárul.

Minden programnak szüksége van arra, hogy adatokat kérhessen be, illetve futásának végeredményét valahol „kiadhassa”. Erre szolgálnak számítógépünk úgynevezett beviteli-kiviteli eszközei, amelyek masinánk nélkülözhetetlen részei.

Beviteli eszköznek számít például a billentyűzet vagy az egér, amellyel alkalmazásaink közvetlenül tőlünk, azaz a felhasználótól kérhetik be az adatokat. Ugyanakkor adatokhoz hozzá lehet jutni például egy állományból vagy egy adatbázisból is, tehát a különböző lemezes egységek is beviteli eszköznek számítanak. Kiviteli eszköz a monitor, a nyomtató, de természetesen egy lemezre is kiírhatunk adatokat, tehát a lemezek egyszerre beviteli és kiviteli eszközök.

Egy többfeladatos operációs rendszerben azonban nem engedhető meg, hogy bármelyik program akkor használhassa a beviteli-kiviteli (a továbbiakban B-K) eszközöket, amikor csak jólesik neki. Ha például két program egyszerre kezd nyomtatni, ennek eredményeként minden bizonnyal valamiféle összevisszaságot kapnánk. Ezért be kell osztani, hogy ki és mikor veheti igénybe az adott eszköz szolgáltatásait. Ez az operációs rendszer feladata.

A másik nehézség, hogy ahányféle B-K eszköz létezik, annyiféleképpen kell programozni. A felhasználói alkalmazások fejlesztői nyilván nem azzal akarnak foglalkozni, hogy alkotásuk milyen színű és formájú gépen lesz futtatva, csupán azt szeretnék, hogy mindenhol fusson, ahol az adott operációs rendszert használják. Ezért szükséges egy olyan eszközfüggetlen környezet megteremtése, amely a felhasználók szeme elől az eszközök zord világát teljes egészében elfedi, és egy egyszerűen kezelhető felületet biztosít. Erről is az operációs rendszer képes gondoskodni.

Azt a részt, amely e két feladat megvalósításáért felelős, az operációs rendszer B-K programjának nevezzük. Sorozatunk következő részeiben ezzel foglalkozunk. Most még csak elméleti síkon, általánosan ismerkedünk meg a B-K eszközökkel, illetve a kezelésükért felelős programmal, a továbbiakban pedig az összes jelentős beviteli-kiviteli eszközön sorban végigmegyünk, a termináloktól kezdve egészen a lemezes meghajtókig.

Egy kis alkatrész...

Ha valamilyen módon csoportosítani szeretnénk a B-K eszközöket, a következő három csoportot állíthatnánk fel: blokkos eszközök, karakteres eszközök és minden egyéb, ami a két másik csoportba nem tartozik bele.

A legjellemzőbb blokkos eszköz a lemez (például merevlemez, hajlékonylemez stb.). Ezek az adatokat előre meghatározott méretű blokkokban tárolják. Minden blokk saját címmel rendelkezik és egymástól teljesen függetlenül olvasható-írható.

A karakteres eszközöktől azonban az adatokat karakterenként egymás után ömlesztve küldjük, illetve fogadjuk, anélkül, hogy bármiféle szerkezeti felépítéssel foglalkoznánk. Klasszikus karakteres eszköz a billentyűzet, a nyomtató, a hálózati csatoló, a modem, az egér és a sort még folytathatnánk.

Ez a csoportosítás nem tökéletes, ugyanis olyan eszközök is vannak, amelyek egyik csoportba sem tartoznak igazán. Nézzük például a szalagmeghajtókat, amelyek nem igazi blokkos eszközök. Esetleg vegyük számítógépünk valós idejű óráját, amelynek az a feladata, hogy meghatározott időközönként megszakítást hozzon létre. Ez is B-K eszköznek számít, mégsem sorolható egyik csoportba sem. Mindezek ellenére a B-K eszközök ilyen módon való csoportosítása jó alapot szolgáltat az operációs rendszer B-K programjának tervezéséhez.

Ezek után nézzük meg, hogyan is épül fel általában egy hagyományos B-K egység. Először is adott egy szerkezet, amely tulajdonképpen nem más, mint maga az eszköz. Ehhez egy elektronikus rész van kapcsolva , amely általában egy nyomtatott áramköri lap. Ez az úgynevezett eszközvezérlő, amelynek feladata a szerkezeti rész irányítása. A vezérlő a számítógép alaplapjához kapcsolódik és a processzorral az úgynevezett sínrendszer segítségével tartja a kapcsolatot.

A hagyományos személyi számítógépek általában az egysínű, más néven buszmodellt alkalmazzák. Ez azt jelenti, hogy a processzor, a memória és a B-K-vezérlők egy sínre vannak felfűzve. Ejtsünk pár szót a vezérlő és a szerkezeti rész kapcsolatáról is. Ez általában nagyon alacsony szintű, ugyanis az eszköz szinte kivétel nélkül mindig bitszervezésű. Ez azt jelenti, hogy a vezérlő a szerkezeti résztől egy bitsorozatot kap, amelyet majd át kell alakítania bájtokká (azaz 8 bites blokkokká). A vezérlő egy belső memóriával is rendelkezik, amelyet átmeneti tárnak (buffer) nevezünk. A fogadott biteket először itt tárolja, majd ha egy bájt összeállt, ellenőrzi annak épségét, és ha minden egybevág, akkor kerülhet tovább a főmemóriába.

A vezérlővel a kapcsolatot az úgynevezett regiszterek segítségével tarthatjuk. Egyes géptípusok esetében ezek a regiszterek a főmemória részét képezik, ezért például úgy adhatunk nekik értéket, hogy a memória egy meghatározott címtartományába írunk (az ilyeneket memóriában leképezett beviteli-kivitelnek nevezzük).

Ha a vezérlő végzett az előírt művelettel és az eredményt elhelyezte a regisztereiben, megszakítást hoz létre. A megszakításokról a folyamatok ütemezésénél már beszéltünk, csakhogy most ki kell egészítenünk néhány dologgal.

PC esetében az alaplapon általában két megszakításvezérlő található. Mindegyik 8-8 bemenettel rendelkezik, de az egyik közülük egymás összekapcsolására szolgál, így egy hagyományos PC-ben 15 megszakítás áll a B-K eszközök rendelkezésére. Ezeket a bemeneteket IRQ-nak (Interrupt ReQuest) nevezzük, mivel ha egy eszköz megszakítást szeretne előidézni, csak elektromos jelet kell küldenie az adott bemenetre. Hogy melyik eszköz melyik bemenethez (azaz IRQ-hoz) kapcsolódjon, azt általában saját maga is eldöntheti, viszont bizonyos eszközök (például a billentyűzet) esetében már előre

be vannak állítva, ezért nincsen lehetőség változtatásra.
Egy megszakításkérés esetében a CPU felfüggeszti azt, amit addig csinált, és az úgynevezett megszakítás-vektortáblázatból kikeresi az adott megszakításhoz tartozó memóriacímet. Ezen a memóriacímen található a megszakítást kiszolgáló eljárás, ami az operációs rendszer része. Ez az eljárás rögvest futni is kezd, és amíg a dolgát be nem fejezi, a processzor nem térhet vissza az eredeti tevékenységéhez.
Bonyolultan hangzik? Sebaj, nézzünk egy példát arra, miként zajlik egy blokk beolvasása a merevlemezről! Először beírjuk a vezérlő regisztereibe, hogy melyik blokkot szeretnénk kiolvasni a lemezről. Miután ez megtörtént, a processzornak nincs több feladata, ettől kezdve a vezérlőn múlik minden. Amíg az dolgozik, az adott folyamat blokkol, és a CPU egy másikat kezd el futtatni. Amikor a vezérlő végzett, megszakítást idéz elő, és az éppen végrehajtás alatt álló folyamat futása felfüggesztődik: ismét a rendszer veszi át az irányítást. Elindul a merevlemez kezeléséért felelős eljárás, amely a vezérlő regisztereiből kiolvassa a művelet végeredményét. Ha a beolvasás sikerült, az eljárás a blokk tartalmát a vezérlő átmeneti tárából bájtonként átmásolja a memóriába.
Mindez eddig rendben is lenne, azonban akad egy gond: a rendszer az adott blokkot kénytelen bájtonként átmásolni, ezzel pedig rengeteg processzoridő megy kárba. Ezért találták ki az úgynevezett közvetlen memóriaelérést, azaz a DMA-t (Direct Memory Access). Ha lehetőség nyílik a DMA használatára, a művelet megkezdése elején a vezérlőnek egy memóriacímet is átadunk, ahová az adott blokk tartalma kerülhet. A másolást majd a vezérlő végzi el, miután a blokk beolvasása megtörtént, és csak ezután hoz létre megszakítást, de a blokk tartalmának bájtonkénti átmásolásával már nem kell foglalkoznunk, mivel az már a memóriában van.
Meg kell jegyeznünk, hogy nem minden számítógép használ DMA-t, sőt, sok esetben egyáltalán nem hatékony, ha a „hagyományos" módszer helyett ezt használjuk. A DMA-vezérlő ugyanis gyakran sokkal lassabb, mint a processzor, és ha a CPU-nak épp nincs semmi más teendője, várakoznia kell, amíg a DMA-vezérlő befejezi a memóriába való másolást; gyorsabban végeznénk, ha ezt programból megvalósított módon tennénk.

A B-K program

A B-K program feladatait a cikk bevezetőjében már ismertettük, most nézzük, hogyan is épül fel. A folyamatkezelés mellett ez is nagyon fontos részét képezi az operációs rendszernek, a rendszermag kódjának nagy részét is ez teszi ki. A legfontosabb elv, amelyhez egy B-K program fejlesztése során ragaszkodnunk kell, az eszközfüggetlenség. Erről már sorozatunk első részében is meséltünk (Linuxvilág 18. szám, 53. oldal), de nem árt, ha szánunk némi helyet arra, hogy egy példa segítségével szemléltethessük, mit is értünk valójában eszközfüggetlenség alatt.
Alapgondolata a következő: az operációs rendszerünk alá fejlesztők szemszögéből teljesen mellékesnek kell lennie, hogy például a bemeneti adatokat tartalmazó állományt merevlemezről, CD-ről, esetleg hajlékonylemezről kívánjuk-e beolvasni. Esetleg az általunk írt MP3-lejátszó kódját ne kelljen csak azért módosítanunk, mert egy olyan gépen szeretnénk futtatni, amely a miénktől eltérő típusú hangkártyával bír. A lényeg az, hogy az operációs rendszer támogassa a használni kívánt eszközöket.
Egy kicsit általánosabban megfogalmazva: például egy `cat <bemenet >kimenet` formájú utasítást is kiadhassunk a parancsértelmezőnek. A `cat` voltaképpen egy program, ami nem tesz mást, mint a bemenetről érkező adatokat a kimenetre folyatja. A lényeg az, hogy a `cat` program szempontjából teljesen mindegy, hogy a bemenet állomány-e vagy valamilyen más fizikai eszköz, például maga a billentyűzet. Az is teljesen mellékes számára, hogy kimenetnek a monitort, a nyomtatót adjuk-e meg vagy egy másik állományt. A `cat` ugyanúgy képes lesz ellátni feladatát, függetlenül attól, mit adtunk meg bemeneti, illetve kimeneti eszköznek.
Az ilyen szintű eszközfüggetlenséget csak úgy tudjuk megvalósítani, ha a B-K programot rétegekbe szervezzük. A legalacsonyabb szinten fekvő réteg van a legközelebb a géphez, és mindegyik réteg egy kicsit többet rejt el a felsőbb szintek elől az eszközből, míg a legutolsó (felhasználói) réteg már csupán egy könnyen kezelhető kényelmes eszközfüggetlen környezetet lát. Ám még mielőtt részletesebben is bemutatnánk a rétegeket, meg kell oldanunk néhány fontos feladatot, például a hibakezelést. Ha egy B-K-művelet közben hibát észlelünk, nem mindegy, hogy melyik rétegben kezeljük. Előfordulhat például, hogy a hiba csak valamilyen ideiglenes állapot miatt következett be (például egy porszem került az olvasófejre), és a művelet megismétlésekor már megkaphatjuk a helyes eredményt. Ezért arra kell törekedni, hogy a hibát az eszközhöz minél közelebbi szinten kezeljük, és a felsőbb rétegek csak akkor szerezzenek tudomást róla, ha a kívánt feladat végrehajtása valóban lehetetlen.
Ha a Linuxot vesszük alapul, a B-K programot négy különálló rétegre oszthatjuk fel: a megszakításkezelőre, az eszközmeghajtókra, az eszközfüggetlen B-K-kódra és a felhasználói rétegre. Legalul a megszakításkezelő foglal helyet. Erről már volt szó a folyamatok ütemezésének tárgyalásakor, de most egy kicsit részletesebben is nézzük meg. Az operációs rendszerek általában arra törekednek, hogy a megszakításokkal foglalkozó eljárások a lehető legalacsonyabb szinten helyezkedjenek el. Amikor egy folyamat B-K-műveletet kezd, az önműködően blokkol. A B-K-művelet elvégzése után az eszköz megszakítást idéz elő, amelyet a megszakításkezelő fogad. Ennek feladata általában csak annyi, hogy felébressze a műveletet indító folyamatot, ami a Linux világában egy üzenet küldésével történik. A megszakításkezelő üzenetet küld a blokkolt folyamatnak, aminek hatására az ismét futásra kész lesz.
Az eszközmeghajtó az operációs rendszernek az a része, ami mindent tud a hozzárendelt eszközről, illetve annak programozásáról. Egyedül ez foglalkozik olyan dolgokkal, mint például a vezérlő regiszterei. Az eszközmeghajtó feladata a következő: az eszközfüggetlen felső rétegekből kapott utasításokat úgymond a vezérlő által emészthető formájúra alakítja át. Ezek az utasítások meglehetősen elvontak, merevlemez esetében például csak annyit kap, hogy olvassa be a 12. blokkot. Ám a merevlemez buta eszköz, csupán nagyon alacsony szintű műveletek elvégzésére képes. Ezért az eszközmeghajtónak az lesz a dolga, hogy megállapítsa, milyen merevlemez-műveleteket kell kiadni, hogy eleget tehessen ennek az elvont utasításnak. Például először ki kell számolnia a 12. blokk fizikai helyét a lemezen, be kell kapcsolnia a fej mozgatómotorját, az olvasófejet a megfelelő helyre kell pozícionálnia és így tovább.
A felsőbb rétegek erről természetesen mit sem tudnak, ők csak a kért blokk tartalmát fogják visszakapni, az összes többi az eszközmeghajtóra van bízva.
Az eszközmeghajtók valójában különálló folyamatok, amelyek a felhasználói szinten futó programokkal együtt ütemeződnek. Igaz, ezek a folyamatok jóval alacsonyabb szinten futnak, és sokkal többet tehetnek, például közvetlenül hozzáférnek az eszközvezérlők regisztereihez, ugyanis az eszközmeghajtók magában a rendszermagban helyezkednek el.

Egy eszközmeghajtó folyamat általában több, egymással szoros rokonságban álló eszközt vezérel. Például létezik egy olyan terminálmeghajtó, amely az összes terminál kezeléséért felelős, de akad olyan ethernetfolyamat is, amely gépünk összes ethernet típusú hálózati csatolójával foglalkozik.
Egy eszközmeghajtó két részből épül fel: egy erősen alkatrészfüggő és egy alkatrészfüggetlen részből. Az utóbbiba azok az eljárások tartoznak, amelyek az eszközmeghajtóhoz tartozó összes eszköznél azonosak. Az alkatrészfüggő részbe pedig csak az kerül, ami kifejezetten ahhoz a típusú eszközhöz használható. Nem kell tehát minden egyes eszközhöz külön eszközmeghajtót futtatni. Az eszközmeghajtók a többi folyamattal a sorozatunk előző részében (Linuxvilág 19. szám, 48. oldal) már ismertetett üzenetküldéses rendszer segítségével tartják a kapcsolatot.
A Linux tehát folyamatokra (és a folyamatok közötti üzenetküldésre) alapuló rendszer rendszer. Ezt a fajta felépítést még a Minixtől örökölte, amely kifejlesztésének az volt a célja, hogy egy jól áttekinthető Unix-alapú operációs rendszert hozzanak létre. Az áttekinthetőbb szerkezet érekében a Unix klasszikus tömbszerű (egy programban megvalósított) felépítését mellőzték, és egy talán jobban átlátható és érthetőbb modellre cserélték.
A folyamat alapú és a tömbszerű rendszer közti különbség a felhasználói és a rendszermagszint közötti kapcsolatban rejlik. Nézzünk például egy fájlból történő olvasást. Az első esetben a felhasználói program üzenetet küld a fájlrendszert kezelő folyamatnak, hogy ő ezt és ezt a fájlt szeretné beolvasni.
A fájlrendszer megnézi, hogy ehhez mely blokkokat kell beolvasni a merevlemezről, majd ő is küld egy üzenetet, amelyben az eszközmeghajtót megkéri az adott blokkok beolvasására. Míg az üzenetet küldő folyamat választ nem kap, addig blokkolt állapotba kerül.
A tömbszerű rendszerekben ilyen nem létezik. Az eszközmeghajtók itt tulajdonképpen eljárások, amelyek a rendszermag által lefoglalt memóriába vannak beágyazva. Ha egy felhasználói rétegben lévő folyamat akar valamit a rendszertől (például egy állomány beolvasását), akkor például egy megszakítás segítségével meghívja a megfelelő magszintű eljárást. Na de itt nem blokkol a folyamat! Ugyanaz a folyamat fut tovább, csak nem felhasználói, hanem rendszermagszinten. Ha a hívott eljárás elvégezte a feladatát, a folyamat visszatér a felhasználói rétegbe. Ne búsuljunk, ha ez egy kicsit zavaros. Egyet jegyezzünk meg: a Linux abban különbözik a legtöbb Unix-alapú rendszertől, hogy folyamatokra alapuló renszer. Ez azzal jár, hogy az egész könnyebben kezelhető kisebb részekből áll, ami az osztott (több gépből álló) rendszerek esetében nagyon jól jön. Akad azonban egy hátránya is: a tömbszerű rendszerek megoldása gyorsabb, mivel egy eljárást kevesebb ideig tart meghívni, mint folyamatok között üzeneteket küldözgetni.
E kitérő után kanyarodjunk vissza eredeti témánkhoz, és vegyük szemügyre a következő réteget, amelyben a B-K program alkatrészfüggetlen része található. Itt olyan tevékenységek zajlanak, amelyeknél lényegtelen, hogy az adott eszköz milyen típusú. Erre a legjellemzőbb példa a fájlrendszert kezelő folyamat. A fájlrendszer felépítése nem változik meg attól, hogy adatainkat valójában milyen lemezen tároljuk, ezzel csak az eszközmeghajtónak kell foglalkoznia.
Milyen feladatokat is kell az eszközfüggetlen B-K-rétegnek ellátnia? Az egyik legfontosabb, hogy minden eszközhöz hozzárendeljen valamilyen nevet, amellyel a felhasználók hivatkozhatnak rájuk. Mint tudjuk, ez Unixban eszközfájlok segítségével történik, amelyek a */dev* könyvtár alatt találhatóak (például a */dev/ttyS0* az első soros kapunkat, vagy a */dev/hda*, amely az elsődleges IDE-merevlemezünket azonosítja). Az eszközfájlok egyébként két számot tartalmaznak, a fő, illetve a másodlagos eszközszámot, amelyek az adott eszközt pontosan azonosítják. Rendszerüknek nemcsak elneveznie, de védenie is kell az eszközöket. Egyrészt meg kell akadályoznia, hogy két folyamat egyszerre ugyanazt az eszközt használja, másrészt azt is figyelembe kell vennie, hogy az adott felhasználó egyáltalán jogosult-e arra, hogy ilyesmit tegyen.
A különböző blokkos eszközök sajnos nem egységes blokkméretekkel dolgoznak. Például nem minden lemeznek ugyanakkora a szektormérete. Ahhoz azonban, hogy fájlokat tároljunk egy fájlrendszerben, elengedhetetlen egy egységes blokkméret alkalmazása. Ezt is itt kell megoldanunk.
A másik feladat az átmeneti tárazás. Előfordulhat, hogy például a felhasználó gyorsabban gépel, mint ahogy a bemenetet fogadó folyamat azt fogadni tudja. Ezért fent kell tartani egy belső tárat, ahová a begépelt karakterek kerülnek, és majd innen kerülnek feldolgozásra a folyamathoz. Aki használt MS-DOS-t, biztos emlékszik rá, hogy miközben a gép dolgozott, és ő nagy lendülettel püfölte a billentyűket, a 16. billentyű lenyomása után a gép vadul sípolni kezdett. Ez azt jelentette, hogy megtelt a billentyűzet átmeneti tára, és addig tele is marad, amíg a begépelteket valaki fel nem dolgozta.
A blokkos eszközök is igényelnek átmenetitár-használatot. Például mi történne akkor, ha egy állomány tartalmát bájtonként szeretnénk beolvasni? Lemezről csak blokkonként olvashatunk. Célszerű tehát az egész blokkot beolvasni az átmeneti tárba, majd kérésre onnan egyenként adagolni a bájtokat.
Vannak még mások is, amik szintén az eszközfüggetlen részben foglalnak helyet, de róluk inkább később, a fájlrendszer tárgyalásakor beszélnénk.
Most már csak az utolsó, a felhasználói szint maradt hátra. Ide a B-K programnak csak nagyon kis része tartozik, az is leginkább könyvtári eljárások formájában. A felhasználói programok ugyanis ezeket az eljárásokat hívogatva végezhetnek B-K-műveleteket, illetve hívhatnak rendszerhívásokat. A könyvtári eljárások tulajdonképpen semmi mást nem tesznek, mint hogy a megfelelő értékeket elhelyezik a rendszerhívásnak megfelelő helyre.
Vannak azonban olyan B-K-eszközök, amelyeket egy másik felhasználói szintű program segítségével használhatunk. Az egyik legjellemzőbb példa erre a nyomtatás. A rendszerben fut egy közönséges felhasználói program, a nyomtatódémon, amelynek az a feladata, hogy a nyomtatást vezérelje. Ha egy folyamat ki szeretne nyomtatni egy állományt, ahelyett, hogy közvetlenül a nyomtatóra küldené, egy meghatározott könyvtárba teszi. Ezután a nyomtatódémon majd eldönti, hogy mikor kerülhet sor az adott állomány kinyomtatására. Ezt a módszert egyébként háttértárolásnak (sorbaállításnak) nevezzük.
Ez a megoldás azért is jó, mert meggátolhatjuk, hogy egy folyamat feleslegesen tartsa fel a nyomtatót (például a nyomtatás késleltetésével), aminek következtében esetleg gátolna más, a nyomtatóra váró folyamatokat.
A háttértárolás másik gyakori felhasználási területe egyébként a hálózaton való állomány-, illetve levéltovábbítás. Ha levelet szeretnénk küldeni, csak elhelyezzük egy megadott könyvtárban, és a levélküldésért felelős démon majd gondoskodik a célba juttatásáról.

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.

Térképek az Interneten

Az ESRI internetes térinformatikai eszközei.

A világ egyik legismertebb térinformatikával foglalkozó vállalata az ESRI . Nem tudok olyan térinformatikai területet mondani, amire a cég ne dolgozott volna ki valamilyen programcsomagot.
Az elmúlt néhány évben az ESRI is nyitott a Linux felé.
A Linuxot elsősorban adatbázis- és egyéb kiszolgálóik alapjául választották, de elkészült már néhány adatfeldolgozást segítő ügyféloldali program linuxos változata is. Most ezek kerülnek rövid bemutatásra.

Az ArcIMS

Ez az az ESRI-programcsomag, ami GIS-megoldások és térképek közzétételét teszi lehetővé, elsősorban az Interneten.
A program neve szó szerinti fordításban Arc internet-térképkiszolgálót (Arc Internet Map Server) jelent. A programcsomag a folyamatábrán (*1. ábra*) látható felépítésnek megfelelően kiszolgáló- és ügyféloldalon egyaránt több alprogramot tartalmaz. A felhasználók és a rendszergazdák számára lehetővé teszi, hogy a térképeket weboldalakon közöljék, rendszerezzék, és ugyancsak a hálózaton keresztül rendszerfelügyeleti teendőiket is ellássák. Jelenleg az ArcIMS Windows- és Unix-(Linux-) rendszereken érhető el.
Az ESRI magyarországi képviseletének köszönhetően a programcsomag kipróbálására is lehetőségem nyílt.
A kiszolgálóoldal épül fel, jóval több elemből áll, mint az ügyféloldal. Két fő része van: a ***Spatial server*** (adatkiszolgáló), amelynek a megjelenítendő adatok és térképek tárolása a feladata, továbbá az adatoknak a szükséges formátumban való szolgáltatása az ügyfélalkalmazások számára. A kiszolgáló egy vagy több virtuális kiszolgáló lehet. Másik része az ***Application server*** (alkalmazáskiszolgáló), amelyre azt is mondhatnánk, hogy a program lelke, hiszen ennek az alkalmazásnak a feladata a különféle térképi adatokhoz kapcsolódó teendők végrehajtása. Ehhez kapcsolódik szorosan a ***Feladatkezelő*** (Tasker) és a ***Feladatfigyelő*** (Monitor). Tennivalójuk az alkalmazáskiszolgáló kiszolgálása, a végrehajtandó munkák irányítása és a feladatok végrehajtásának figyelése, illetve ellenőrzése.
Nézzük meg egy kicsit részletesebben is a programot:

- Az ***ArcIMS Monitor*** az ArcIMS adatkiszolgáló működését figyeli. Ha a rendszer újraindul, az ArcIMS Monitor önműködően újraindítja a szolgáltatást.
- Az ***ArcIMS Tasker*** törli az ideiglenes képfájlokat, amelyeket az ***Image Services*** (képszolgáltatás) hoz létre a felhasználó által megadott időközönként.
- ***ArcIMS Connectors*** (ArcIMS kapcsolatkezelő) és ***Web Server*** (webkiszolgáló): ez a két eszköz szervesen összetartozik. Feladatuk az ArcIMS alkalmazáskiszolgáló és az Internet vagy a kiszolgálandó egyéb hálózat közötti kapcsolat létrehozása.

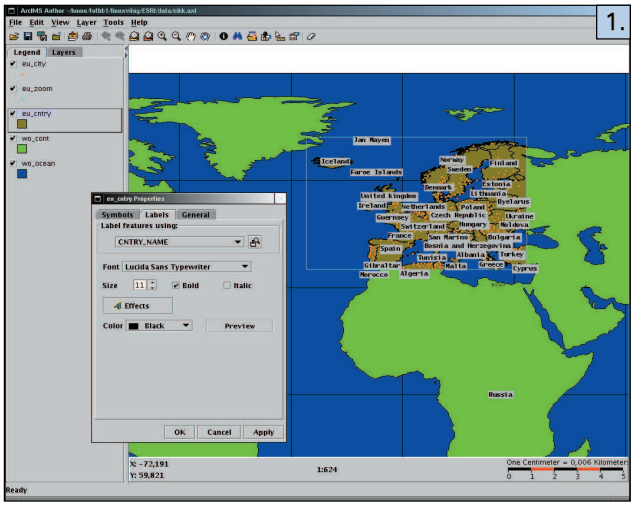
Az alkalmazáskiszolgálónak négy típusa van:

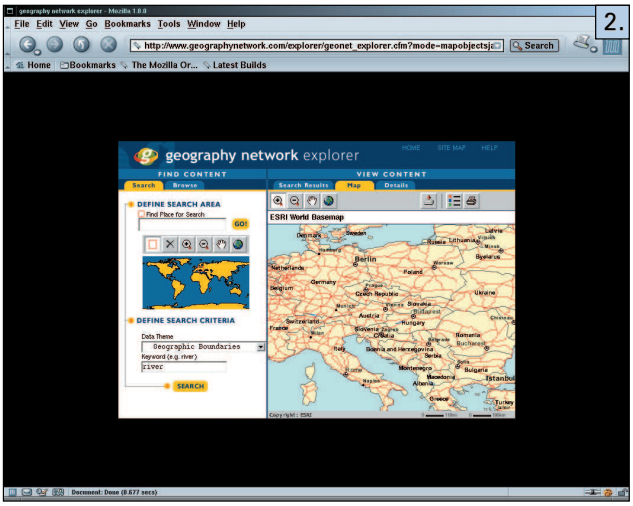
- ***ArcIMS Servlet Connector***: az ESRI saját megoldása.
- ***ColdFusion Connector***: saját ColdFusion-ügyféllel működik.
- ***ActiveX Connector***: csak ActiveX-ügyféllel működik.
- ***JSP Connector***: Java-alapú megoldás.

A webkiszolgálói oldalon több eszközt is használhatunk: az ***iPlanet***-et, az ***Oracle Application Server***-t, a ***Tomcat for Apache***-t, a ***WebLogic***-ot és a ***WebSphere 3.5.5***-t.
Ezeknek az eszközöknek bármelyikéhez kapcsolódhatnak a különféle grafikus felületű felügyeleti, valamint lekérdező és megjelenítő eszközök.

- ***ArcIMS Manager*** (ArcIMS felügyeleti eszközök): az ArcIMS-csomagban olyan előre elkészített beállítófájlok, megjelenítő- és szerkesztőeszközök találhatók, amelyek akár helyi gépről vagy távolról is lehetővé teszik a rendszer irányítását és karbantartását. Ezeknek az eszközöknek a többsége ránézésre szokásos weboldalnak látszik, egy részük viszont Javával támogatott környezetben válik használhatóvá.
  Az ***ArcIMS Manager*** a következő alkalmazásokat tartalmazza: ***Author***, ***Administrator*** és ***Designer***.

Az ***Administrator*** feladata a weboldallal kapcsolatos beállítások végrehajtása és a karbantartás, továbbá az ***Author*** és a ***Designer*** számára grafikus indítási lehetőséget ad.
Az ***Author*** feladata az alkalmazások beállítófájljainak karbantartása, valamint ezzel az eszközzel tudunk új szolgáltatásokat is létrehozni. Az általa készített beállításokat tartalmazó fájlok határozzák meg a térképi tartalmat. Ezenkívül megadja, hogy mely rétegek és hogyan kerüljenek megjelenítésre (szín, jelkulcsok, feliratok stb.).
A ***Designer*** feladata a weboldal megjelenésének szerkesztése.
Az ArcIMS része még az ***ArcIMS viewer***, amely weblapalapú megjelenítő eszköz; feladata elérhetővé tenni a térképszolgáltatásokat és a böngészési, elemzési szolgáltatásokat az ügyfél webböngésző felületén.
Az ***ArcMap Server*** egy ugyancsak könnyen használható ún. ***ArcIMS Virtual Server*** (ebben az esetben „virtuális” alatt azt értjük, hogy ugyanazon a gépen, egy és ugyanazon ArcIMS-kiszolgálón újabb szolgáltatásként is futhat), ami az ügyféloldali felhasználó számára lehetővé teszi, hogy az ***ArcMap*** vagy az ***ArcIMS Author***, illetve az ***ArcIMS Manager*** segítségével térképet hozzon létre és megjelenítse őket az Interneten.
A lehetőségekhez mérten otthoni, egygépes környezetben megpróbálkoztam az ArcIMS beüzemelésével. A linuxos változat még Red Hat 7.1-re készült. Ennek meg is lett a böjtje. A telepítő parancsfájl első futtatásánál legtovább a „nem együttműködő” Debian (SID, július végi változat) alatt jutottam. Red Hat (7.3-as változatszámú, a letölthető háromlemezes változat) alatt a szükséges csomagok egyáltalán nem álltak rendelkezésre; ha mégis elérhetőek voltak, visszafelé akkor sem voltak cseresabatosak. Ezek után a telepítést Debian alá visszatérve folytattam. A fenti webkiszolgáló listából a Debian az Apache-Tomcat párost tartalmazta, ezért ezt választottam a telepítéshez is. A hivatalos leírás egyébként a letölthető *.tgz-változatok telepítését taglalja. Jó, ha rendelkezünk a JRE- és esetenként a JDK-csomagokkal is, hiszen a telepítés egyik előfeltétele a megfelelően beállított `JAVA_HOME` környezeti





változó és az előzőleg telepített Netscape (ez utóbbit nálam – egy hivatkozás segítségével – a Mozilla helyettesítette). Nincs grafikus telepítő és a telepítő parancsfájl sem működött megfelelően Debian alatt. Ezt azonban könnyen áthidalhatóvá teszi az ESRI ArcIMS leírása, ami enyhén szólva részletesen, lépésről lépésre segíti végig a rendszergazdát a telepítési folyamaton. Véleményem szerint azonban a telepítés folyamata meglehetősen sok buktatót tartogathat (főleg, ha először csinálunk ilyesmit).

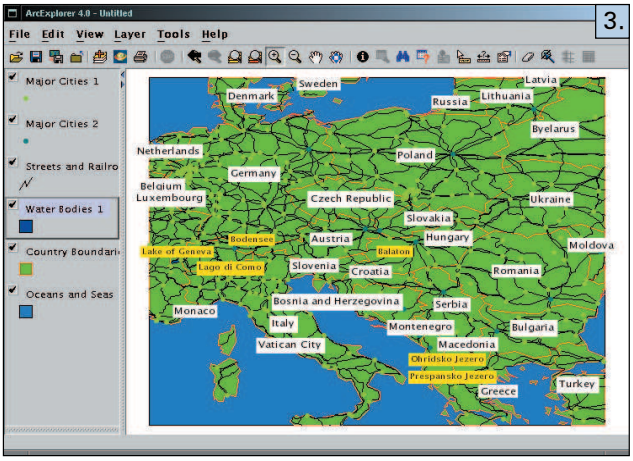
Néhány kellemetlen óra után a kiszolgáló végre elindult. Ezt követően először a kiszolgálóoldali grafikus eszközöket próbálgattam, hogy vajon melyik mire képes. Az eszközök mindegyike javás környezetet igényel a futtatáshoz. Sajnos a Java-változatok sem teljesen csereszabatosak visszafelé, először ugyanis az 1.4-es JRE-vel próbálkoztam, de megjelenítési nehézségek adódtak. A CD-n szereplő 1.3-as Java-változatot beállítva a helyzet már jelentősen javult.

Nos, az alkalmazásfelügyeleti eszközök közül részletes bemutatásra csak az ***Author***-t tartottam alkalmasnak, hiszen a többi tényleg csak a kiszolgáló szigorúan vett karbantartási feladatait látja el, a beállítások pedig az adott kiszolgáló feladatának megfelelően szintén igencsak egyediek lehetnek.

ArcIMS Author

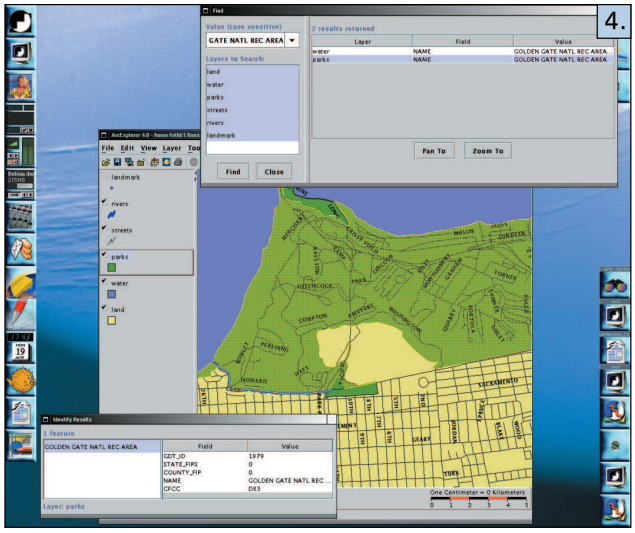
Az Author elsősorban megjelenítési feladatok ellátására készült, nem szerkesztési munkák végrehajtására. Ennek megfelelően a program csak az adatok rétegsorrendjének, formátumának beállítására alkalmas eszközökkel rendelkezik (*1. kép*).

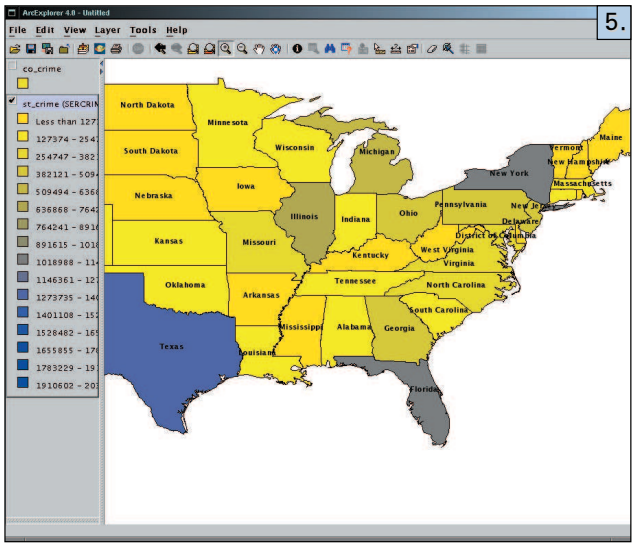
Az ablak tetején szokásosan a menük, illetve a különféle eszköztárak találhatók. Az ***Author*** csak a már elkészített térképeket képes feldolgozni. Két alapvető lehetőségünk van az adatok beolvasására. Az első, hogy egy projektfájlt nyitunk meg, aminek az lehet az előnye, hogy ebben már előre összerendezett rétegeket találhatunk. A projektfájl kiterjesztése ***.axl***. A másik lehetőség a rétegek egyesével történő kiválasztása és betöltése. Várhatóan ez a több munkát igénylő feladat. A rétegek két helyről olvashatók be: a helyi gépről vagy ***ArcSDE*** kiszolgálóról. Szerencsére rendelkezésemre állt egy windowsos ***ArcExplorer*** bemutatópéldány is, ami előre összerendezett térképi rétegeket tartalmazott (a ***.shp***-fájlok tartalmazzák a grafikus adatokat, a ***.dbf***-adatbázisfájlok a leíróadatokat, amelyek a grafikus térképi elemekhez vannak rendelve). Az így beolvasott rétegek a ***Legend*** (jelmagyarázat) ablakban jelennek meg a beolvasás sorrendjében. A lista első eleme a rajz legfelső



rétege. Ez annyit jelent, hogy a listában alatta szereplő rétegek a térképen felső kiterjedésének megfelelően takarásba kerülnek. Az ablak bal oldalán találjuk a rétegek, illetve a jelmagyarázat beállítására szolgáló sávot. Az egyes rétegek mellett szereplő jelölőnégyzetek segítségével tudjuk az egyes rétegek megjelenítését ki-bekapcsolni. Ha a jobb egérgombbal a réteg nevére kattintunk, a megjelenő menüben több beállítást is el tudunk végezni: például réteget törölhetünk a listából, beállíthatjuk a legkisebb-legnagyobb megjelenítési méretarányt, valamint a megjelenítési tulajdonságokat (szín, betűtípus stb.) is. A rétegek sorrendje az előbbi felsorolás használatával vagy egyszerűen nevének az egérrel történő kiválasztásával és megfelelő helyre húzásával változtatható meg. Az előbbi felsorolásban szereplő „legkisebb-legnagyobb megjelenítési méretarány” beállításra visszatérve: a kifejezés azt jelenti, hogy egy felirat például 1:10 000-es méretarányban még nem, 1:100 000-es méretarányban pedig már nem jelenik meg, csakis a két méretarány között. Erre azért van szükség, mert a térkép méretarányával arányosan nem változik a feliratok mérete, és ez nagy méretarányban (1:10 000) még csak szükségtelen további adatot jelenthet, kis méretarány esetében (1:100 000) azonban maga a felirat takarhat le számunkra értékes adatokkal bíró nagy területeket.

Az egyes rétegek további megjelenítési beállításaira a ***Layers*** fülön belül a megfelelő réteg kiválasztása után nyílik lehetőségünk. Ezen túl mód kínálkozik több réteg egyszerre történő megváltoztatására is, illetve az egyes rétegelemek megjelenítési sorrendjének módosítására is.


4.


5.

A programban használható néhány érdekesebb szolgáltatásról röviden:

- Egyéni mértékegységekkel megjeleníthető méretarány-vonalzó, amelyen fokok, angolszász mértékegységek és metrikus értékek adhatók meg mértékegységként.
- Adatok az egyes rajzelemekről – az eszközt használva a lekérdezendő rajzelemre kattintva a hozzá kapcsolódó adatbázisadatokat megjeleníthetjük egy táblázatban.
- Keresésszolgáltatás, amelyben a megfelelő – például – helységnév megadásával megjeleníthetünk egy listát, ebben lesz látható a keresés eredménye. Ha ebben a listában kiválasztunk egy-egy sort, a hozzá tartozó területre rá tudunk állni, illetve nagyítani tudjuk.
- Előre tárolt lekérdezés létrehozásának lehetősége – módunkban áll az itt elkészített lekérdezés elérése az általunk elkészített weboldalon szereplő keresési listákban. Változók használata is lehetséges.
- Geokód-tulajdonságok beállítása. A geokód röviden egy adott térképi elemhez rendelt egyedi azonosító, ami csakis egy elemre vonatkozhat (egy térképi elemre egyetlen adatbázisrekord mutat). Idézném a program súgójának példáját: egy megjelenítendő térképi terület két várost tartalmaz, amelyek mindegyikében létezik egy Fő utca. Annak érdekében, hogy az utcák egyértelműen azonosíthatóak legyenek, a névhez egy másik azonosítót rendelünk hozzá, például a település nevét. A hozzárendelt értéken túl még meglehetősen sok további geokódra vonatkozó változó beállítását is lehetővé teszi a szolgáltatás.
- ***Maptips*** szolgáltatás, amelynek feladata hasonló a programok ikonsorai fölé húzott egérmutató esetén megjelenő eszközsúgóéhoz. Ennek használatával egy térképi elem felett rövid leírószöveget jeleníthetünk meg.

A program a felsoroltakon kívül nem tartalmaz túl sok szolgáltatást (a térkép eltolása, a méretarány változtatása, nyomtatás, a kész térkép kimentése ***.jpeg*** fájlba).
Érdekes hibát találtam a program kipróbálása során: ha a hozzárendelt szöveg az adott területen nem fér el, akkor esetenként elég megtévesztő módon áthelyezésre kerül (a Russia felirat a betűméret növelésével Afrika közepén kötött ki).
Összességében a program elfogadható sebességű (a Javának köszönhetően), és a kevésbé hozzáértők számára is egyszerű és könnyen használható.

ArcExplorer

Az ArcExplorer feladata az ügyféloldali adatmegjelenítés igényeinknek megfelelő módosítása. A program ránézésre nem sokban különbözik az előbb ismertetett ***ArcIMS Author***-tól, hiszen szinte minden szolgáltatásukban azonosak. Különböző feladatuk miatt csak egy-két kisebb eltérés akad a két program között. Ezek közül néhányról röviden is szólnék:

- Térképeket tölthetünk le internetes adatbázisokból és az ***ArcExplorer***-rel megnyithatjuk őket. Az adatokat a rétegek hozzáadásánál található ***Geography Network*** gombbal elindított Netscape (nálam Mozilla) segítségével tölthetjük le a ➲ http://www.geographynetwork.com/explorer címről (*2. kép*).

A letöltött zipfájlt kicsomagolva a rétegeket a szokásos módon olvastathatjuk be. Fontos megemlíteni, hogy csak az a terület és annak adatai kerülnek letöltésre, amit a weboldal előnézeti térképén kijelöltünk (*3. kép*).

- A térképek adatai alapján lehetőség nyílik egyedi lekérdezések létrehozására (*4. kép*).
- Lehetőség nyílik rajzelemek kiválasztására, akár több lépésben is, valamint távolságmérésre az általunk megadott mértékegységnek megfelelően.

Az ***Author***-ra is jellemző, hogy a rétegeket valamilyen tematika szerint több színnel is meg lehet jeleníteni (*5. kép*).
Ezt a rétegtulajdonságoknál tudjuk beállítani, csak meg kell adnunk, hogy mely két szín közti árnyalatok jelöljék az eltérő tulajdonságú területeket.
Mint említettem, én csupán az ArcExplorer windowsos változatával rendelkezem. A két változatot összehasonlítva és a Linux alatti Java-alap okozta időnkénti (egyébként elenyésző) lelassulásoktól eltekintve azonos minőségű megoldást jelenthet az ArcIMS-felhasználók számára.
Remélhetően a későbbiekben további ESRi-termékekkel találkozhatunk Linuxon is.

Tóth Béla
(tothb1@freemail.hu) Nős, két gyermek büszke atyja. Dolgozott földmérőként, majd térinformatikus szakmérnöki képesítést szerzett. Egyaránt otthonosan mozog a CAD és a térinformatikai programokban, valamint a DOS- és Windows-alkalmazásokban. Legkedveltebb elfoglaltsága már két és fél éve a Linux.

Egy másik környezet

Vegyünk egy gyakori élethelyzetet:
Hubának megtetszik a Linux, és úgy dönt, hogy kipróbálja.

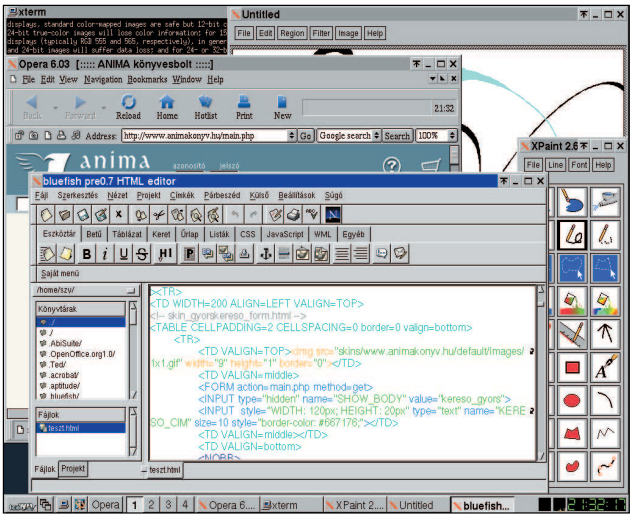
Főhősünk bizonyos fokig lusta, ezért az első lépéseknél megkéri a barátját, hogy segítsen neki telepíteni a rendszert. Ez sikeresen meg is történik, csupán a magmodulok körül volt egy kis hézag, valamint a videokártya belövésénél. Amikor a Linuxszal való ismerkedés első lépésein átverekedte magát, majd végigrágta az egyik bevezető szintű linuxos könyvet (például a Pere-féle Felhasználói ismereteket), boldogan ül a gép előtt, majd rájön, hogy csuda jó a Linux, de mire használja? Tegyük fel, hogy fejébe veszi, márpedig ő weblapokat akar fejleszteni egy kisebb teljesítményű gépen. Milyen programokra van szüksége?
Nézzük végig... A rendszer már üzemel és az internetkapcsolat is megvan. Kell egy gyorsabb ablakkezelő, ami nem feltétlenül fordítja a gép erőforrásainak döntő többségét a háttérben vonagló női aktok megjelenítésére, egy HTML-szerkesztő, egy grafikai program, valami, aminek a segítségével kedvenc fotóalbumait készíti el és így tovább.

A munkakörnyezet

Huba ismeri a KDE-t, a Gnome-ot, valamint ***Marcel Gagné*** Kicsi, de jól működő Linux-doboz című cikkének (78. oldal) valamint az 59. oldalon lévő IceWM második pillantásra című cikk elolvasása után az IceWM és a BlackBox ablakkezelőkről is hallott. Sőt, fellelkesedett, és kipróbált még rengeteg ablakkezelőt, majd rájött, hogy igazából nincs szüksége semmilyen különleges szolgáltatásra: szeretné, hogy ablakai között könnyen mozogjon, valamint hogy a használni kívánt programjait gyorsan és kényelmesen el tudja indítani.
Huba itt egy érdekes kérdésbe futott bele. Értette, hogy létezik egy rendszermenü, ami mindegyik ablakkezelő alatt elérhető, de honnan szedi ezt a menüt a rendszer, és hogyan alakíthatná ki Huba, hogy saját menüje legyen? Leegyszerűsödne az élete: egy külön menüpontba kirakná a néhány programot, ezek után az asztalon a jobb gombot nyomva vígan elérhetné az összes kedvencét. Huba gépén GNU/Debian csücsül (Miért? Csak.), és addig kutakodott a menük kérdésében, amíg meg nem találta a ***menu*** csomagot. Röviden összefoglalva Huba teendője annyi, hogy a megfelelő szerkezetű fájlt a ***/etc/menu*** alá helyezi, majd az `update-menu` parancsot adja ki. A „megfelelő szerkezet”-re példát a ***/usr/lib/menu*** alatt, leírást pedig a ***/usr/doc/menu*** alatt találhatunk.
No de lépjünk tovább. Huba végre megint eltölthet néhány napot a rendszerrel való ismerkedéssel, majd kiválasztja a neki legjobban tetsző ablakkezelőt. Rendben, csakhogy annyira hozzászokott, hogy a háttérben állandóan zene szól, és a KDE alatt kényelmes volt a sok grafikus kütyü. Zenedoboznak mindenhol megteszi az `xmms`, ha pedig keverőpultot keresünk, egy egyszerűbb, akár karakteres felületű programot is használhatunk (nem olyan szép, de legalább nem kell alá egy halom könyvtár), például az `aumix` programot vagy a `mp3blaster` csomagban található `nmixer`-t.
Ezzel teljes a munkafelület, kezeli az ablakait, könnyedén eléri programjait, szól a zene, mi kell még a jó munkához?

Képek kezelése

A weboldalakon általában előfordul egy-két kép. Kezelnünk kell tehát valahogy a képekkel kapcsolatos igényeket is. Természetesen első helyen említhetjük a Gimpet (illetve a Gimp 1.2-t), ha azonban erre nincs szükség, akkor is találhatunk egy seregnyi hasznos, kevésbé okos programot. Huba rendszeresen olvassa a Linuxvilágot, így tudja, hogy az elérhető csomagok között könnyedén keresgethet, a keresgéléshez csupán a rendszere által használt csomagformátum szerint kell választania egy segédeszközt. Mivel GNU/Debian alatt dolgozik, elindítja az `aptitude`-ot, majd egy kategória szerinti



nézetben a `Graphics programs::Editors` alatt körbenéz, kipróbálgat néhányat, végül az `xpaint` mellett marad – az általa elvégzendő egyszerű igényeknek a program által biztosított képességek tökéletesen megfelelnek.
Leendő webfejlesztőnk két kiegészítőprogramot is használ, képnézegetőnek a `gqview`-t (azt már megszokta, ismerősei komoly erőfeszítése ellenére is, többen barátai közül ugyanis `showimg`-t használnak), valamint a felhasználóoldali képtérképek (amikor a weboldalon egy képen belül több érzékeny területet is megadhatunk) elkészítéséhez az `imaptool`-t.
Még egy érdekes kérdés, hogy mivel tudja megfelelő formátumra alakítani a nyers képeket, illetve például albumok esetében, hogyan tudja könnyedén az összes képet kicsinyíteni, mondjuk 750 képpont szélességűre. Ehhez a talán legegyszerűbb (és a héjprogramokban is használható!) program az `ImageMagick` csomag `convert`-je, ennek leírásában (`man convert`) már a legelején egy igen jól használható példát kapunk:

```
convert -size 120x120 cockatoo.jpg -resize
➥120x120 +profile '*' thumbnail.jpg
```

Remek, akkor kezdődhet a komoly munka! Hubának kell még egy HTML-szerkesztő, meg valami ügyes kis eszköz, amivel a munkáját könnyedén közzéteheti. Valamint nem lenne rossz, ha azokat a fránya fotóalbumokat is egyszerűen el tudná készíteni.

Webes munkák

Az első, ami Huba számára elengedhetetlen, egy olyan eszköz, amivel a HTML-oldalakat szerkesztheti. Természetesen – mint tudjuk – a webszerkesztő nagymestereknek elég egy `vi` vagy egy `mcedit` is, mi azonban ennél azért kényelmesebb munkaeszközre pályázunk. A HTML-szerkesztők között is széles a választék. Mivel Huba sok jót hallott róla, először a `bluefish`-t próbálta ki, és mivel tökéletesen megfelelt igényeinek, meg is tartotta.

A következő nagy kihívás a fotóalbumok elkészítése. Huba azt gondolta, először ír majd egy kis héjprogramot, ami a képeket összegyűjti az adott könyvtárból, mindegyikhez készít egy-egy HTML-fájlt, amelyben leírásokat fűzhet a képekhez, majd egy fotóalbumoldalt is, ami a képek előképeit tartalmazza. Ezzel a kis héjprogrammal azután egy-egy könyvtárból könnyedén készíthet albumot. Már meg is örült, hogy végre megismerkedhet a héjprogramozás rejtelmeivel, nagy bőszen nekilátott ***Büki András*** új Unix/Linux héjprogramozás könyvének, amikor eszébe jutott, hogy ezt a héjprogramot bizony sokaknak meg kellett írnia. Lehet, hogy valaki épp egy olyan programot készített, ami pont ezeket az igényeket elégíti ki? Így akadt rá a `cthumb` csomagra.

Hogy mire is jó a `cthumb`? Két üzemmódot ismer, az első esetben a megadott fájlokhoz egy albumfájlt készít, a másodikban pedig az albumfájl feldolgozása közben elkészíti az előképeket, valamint a szükséges HTML-fájlokat.

Akkor hogyan is készüljenek az albumok? Először a képeket a fotóalbum számára egy külön könyvtárba le kell kicsinyíteni (Huba a 750 képpontos szélesség mellett döntött), majd a könyvtár képeiből egy albumot készít, végül az albumot feldolgozva létrehozza a HTML-fájlokat. Később a képekhez tartozó szöveget nyugodtan beleírhatja a képhez tartozó HTML-fájlokba (bízva benne, hogy nem törli őket véletlenül, ha egy új kép miatt újrakészítteti az albumoldalt).

De mint tudjuk, Huba lusta, ezért addig rágta a barátja fülét, amíg szegényke beadta a derekát, és két héjprogramot készített. Az első feladata, hogy az adott könyvtárban lévő fájlokból egy új könyvtárba készítse el az egységes méretű képeket. Ez sikeredett egyszerűbbre:

```
#!/bin/bash

mkdir album
for i in `ls *jpg`
  do
   if   [ -e album/$i ]
    then
       echo " * album/$i mÆr lØtezik"
    else
       echo "album/$i kØsz tØse"
       convert -size 750x562 $i -resize
       ➥750x562 +profile '*' album/$i
   fi
  done
```

A programocska nem túl okos, helyben elkészít egy alkönyvtárat, majd belepakolja a lekicsinyített képeket. Ezután Hubának az albumkönyvtárat át kell másolnia honlapja területe alá egy megfelelő névvel, majd a két alábbi parancsot szükséges futtatnia:

```
cthumb -c *jpg >cthumb.album
cthumb -m -x 200 -y 150 cthumb.album
```

A `-m` kapcsoló hatására a `cthumb` nem készít ***index.shtml*** fájlt (nekünk erre semmi szükségünk), a `-x` és `-y` kapcsolókkal pedig az albumoldalon megjelenő előképek méretét állítottuk be. Ha például az albumokat tartalmazó könyvtár a honlapon az ***albumok***, könnyen hivatkozhatunk egy-egy albumra: `<a href="albumok/szulinap/page.html">Sz linapi buli</a>`. Természetesen a megoldás nem tökéletes, Hubának még készítenie kell egy fejlécet, amit az összes HTML-fájl elejére berakat majd a `cthumb*`-bal, hogy az oldalak külalakja egységes legyen, valamint hogy az egyes albumokból vissza tudjon lépni a honlap megfelelő helyére. Huba viszont továbbra sem nyugodott, amíg meg nem kapott egy olyan héjprogramot, ami az ***albumok*** könyvtárból indítva el nem készíti neki az összes albumot (ha a képekhez már léteznek a HTML-fájlok, akkor ne törölje azokat stb.). Huba feladata csak annyi, hogy az elkészítendő album könyvtárának nevét egy `createall` után beillessze a fájl végére.

```
#!/bin/bash
#
#   az   sszes albumot elkØsz ti

BASEDIR=`pwd`

function createall () {
    cd $1
    echo "************************************"
    echo `pwd`
    echo "------------------------------------"
    if [ -e cthumb.album ]
       then
            echo "INFO     **** cthumb.album
            ➥lØtezik, nem kØsz tek œjat ****"
       else
            rm *thumb*jpg #elıkØpek t rlØse
            rm *html      #webolapok t rlØse
            cthumb -c *jpg >cthumb.album
       fi
    cthumb -m -x 200 -y 150 cthumb.album
    cd $BASEDIR
}
createall szulinap
createall balaton
createall csalad/anyu
createall csalad/nagyiszulinapja
```

Huba tehát boldog, mivel a feladathoz szükséges összes eszközt könnyedén használja, és még azon a buta öreg gépen is tűrhető sebességgel tud dolgozni.

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

Debian Woody 3.0

A nagy visszatérés?

Cikkünk elején egy kis Debian/GNU Linux-történelemórát tartunk, nyomon követjük az előző és a mostani megbízható változat megjelenési időpontjait.

- 2000. augusztus 15. – A Debian Projekt örömmel jelenti be a legújabb változat, a 2.2-es (kódnevén Potato) megjelenését.
- 2002. május 1. – Az újabb változat megjelenésének beígért dátuma, mialatt a Potatót még mindig fejlesztik. Ekkor a hivatalos Debian, a 2.2r6-os jelenik meg. A 3.0-s (kódnevén Woody) kiadása csúszik, mert még 11 biztonsági foltot kell alkalmazni a terjesztésen, mielőtt üzembiztossá válhat. Ezen már senki sem lepődik meg, viszont egyesekben felmerül a kérdés, hogy lesz-e egyáltalán üzembiztos Woody.
- 2002. július 13. – Megjelenik a 2.2r7. A fő változatszám után álló r7 a „revision 7” angol kifejezés rövidítése, és azt jelöli, hogy az adott változatnak hányadik kiadásáról van szó.
- 2002. július 19. – Igen, itt van. Közel kétéves várakozást követően megérkezett a megbízható Woody.

Woody, a megbízható

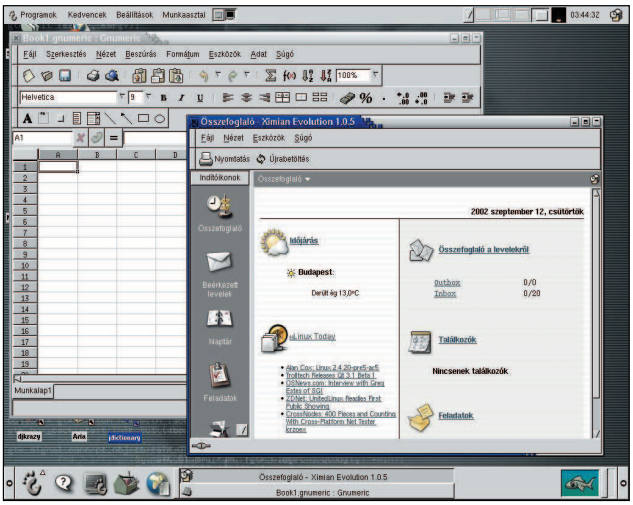
A Debian GNU/Linux 3.0 (Woody) összesen 11 processzorcsaládot támogat: a kézi gépektől egészen a nagy kiszolgálókig, beleértve a legújabb, 64 bites processzorokat. A terjesztés hét CD-t foglal magában processzorcsaládonként. Számos ablakkezelőt tartalmaz, a két leghíresebb közülük a KDE 2.2.2, illetve a Gnome 1.4. Az XFree86 4.1-es változata található meg benne. Az alapcsomag része a GNU Privacy Guard (GPG), ahogyan az OpenSSH is. A Woody megfelel annak az ajánlásnak, amely a fájlrendszer faszerkezetének alapjait írja le, név szerint az FHS 2.2-nek (Filesystem Hierarchy Standard). A 2.2-es vagy 2.4-es sorozatú rendszermag is használható. A 2.4-es sorozat esetén lehetőség nyílik ext3 fájlrendszer, illetve ReiserFS fájlrendszer használatára. A többnyelvű telepítő segítségével a rendszer többek között magyarul is telepíthető.

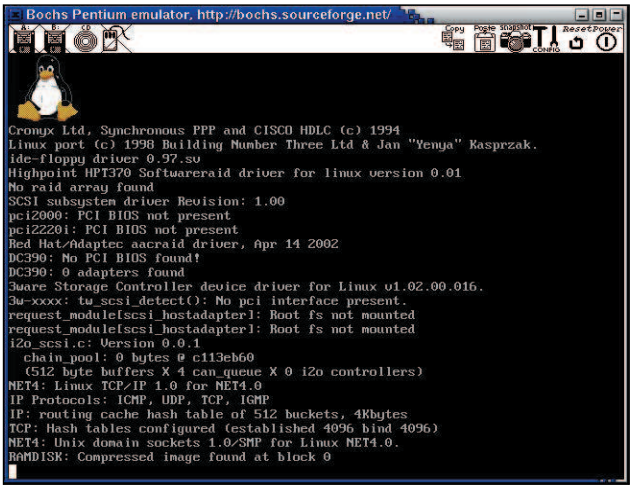
A jigdo

Amikor egy Debian-kiadás napvilágot lát, a megfelelő ***.deb***-csomagokat felteszik az Internetre, ám a CD kiírásához szükséges ***.iso***-ra még várni kell egy-két hetet. Ezért próbáltam ki a jigdót, amellyel még a hivatalos lemezlenyomat előtt elkészíthettem a sajátomat. A jigdo (Jigsaw Download – képkirakós letöltés) egy olyan eszköz, amivel a már említett csomagokat le lehet tölteni, majd a lemezlenyomatot felépíteni, végül a CRC ellenőrzőösszeget az eredetivel összehasonlítani. Mindezt egy parancssorral, ráadásul gyorsabb, mintha az egész lemezlenyomatot kellene letölteni, mert lemezlenyomat-tükörből jelenleg kevés akad, és azok is nagyon le vannak terhelve. Egyszerű használni, mivel minden szükséges adatot egy ***.jigdo*** állomány tartalmaz, ennek a címét kell megadni, majd hátradőlve figyelni, ahogyan a lemezlenyomat felépül. Sőt, még a korábban megszakított letöltésekhez is vissza lehet térni. A jigdónak windowsos és solarisos változata is létezik, így a lemezlenyomatot más operációs rendszer alatt is el lehet készíteni.
A Debian honlapján az áll, hogy a legújabb jigdót kell használni, a korábbi változatokkal a letöltés nem működik. A legújabb jigdo a 0.6.8-as, amely még a kipróbálás alatt álló Woody-változatban sem volt megtalálható. Így előbb muszáj a legfrissebb jigdót letölteni, majd telepíteni. A honlapon és a CD-mellékleten (➲ http://home.in.tum.de/~atterer/jigdo/, Magazin/Woody) megtalálható a ***.deb*** csomag, a windowsos, és a solarisos változat, illetve a forráskód. Én a ***.deb*** csomagot telepítettem. A ***http://home.in.tum.de/~atterer/jigdo/jigdo-file_0.6.8-1_i386.deb*** állományt töltöttem le, majd az alábbi a parancsot adtam ki:

```
# dpkg -i jigdo-file_0.6.8-1_i386.deb
```

Egyébiránt előbb a terjesztésben található változatot érdemes telepíteni, majd így frissíteni, mert ezzel a módszerrel a jigdo telepítésénél nem kell figyelni a függőségekre. Többek között a `wget` -re is szüksége van, mert ezt használja a letöltéshez. Telepítés után a rendszer két fontos futtatható állománnyal bővült: a ***jigdo-file*** és a ***jigdo-lite*** nevűekkel. A ***jigdo-lite*** a ***jigdo-file*** lebutított változata, amivel semmi gond nincs, én is ezt





használtam, mivel az alapértelmezett beállításokat szükségtelen volt átállítani. A párbeszédre épülő letöltésvezérlőt egyszerűen a következő paranccsal el lehet indítani:

```
# jigdo-lite
```

A ➲ http://non-us.cdimage.debian.org/jigdo-area/current/jigdo/ alatt találhatók a ***.jigdo***-állományok, processzorcsalád szerint rendszerezve. Az ***i386***/ könyvtár alatt nyolc ***.jigdo*** fájl van, mivel az első Woody CD-nek létezik ***us***, illetve ***non-us*** változata. Ez az Egyesült Államokban hatályban lévő titkosításra vonatkozó megszorítások miatt van így. A két CD csupán annyiban különbözik, hogy a ***non-us*** egy-két csomaggal többet tartalmaz – érdemes az utóbbit választani. Ezek alapján a ***jigdo-lite***-t hétszer kell majd elindítani, és egyesével a következő ***.jigdo***-állományokat kell neki megadni:

```
woody-i386-1_NONUS.jigdo
woody-i386-2.jigdo
woody-i386-3.jigdo
woody-i386-4.jigdo
woody-i386-5.jigdo
woody-i386-6.jigdo
woody-i386-7.jigdo
```

Természetesen a ***jigdo-lite***-nak a teljes címet kell megadni. Egy-két egyszerű kérdés után el is kezdődik a letöltés. Helyi tükörnek én az ➲ ftp.hu.debian.org helyet adtam meg.

Találkozás egy régi egeremmel…

A CD-k kiírása után izgatottan tettem be az első lemezt a meghajtóba. A számítógép elindult, majd a Debiannál már megszokott üdvözlő üzenet fogadott. A Woody mellől végre eltűnt a ***testing/unstable*** felirat. Az ENTER leütésével kezdetét veszi a telepítés – az alapértelmezett rendszermaggal, ami 2.2-es sorozatú, ezért mielőtt vadul telepíteni kezdenél, F3-mal érdemes egy pillantást vetni a használható rendszermagokra. A `bf24`-et beírva a telepítő már a 2.4.18-as rendszermaggal indul. Az első meglepetés a nyelvválasztásra felszólító képernyőn ért, amikor megláttam egy olyan sort, ami magyarul szólt. A telepítőrendszer ettől fogva hellyel-közzel magyar nyelvű. A már megszokott felépítés mellett most végre magyar nyelvűek a feliratok. Mindössze egyetlen új ablakkal találkoztam. 2.4-es sorozatú rendszermag esetén a ***Linux-partíció inicializálása*** menüpont alatt a megkérdezi a használni kívánt fájlrendszer típusát. Lehetőség van ext2, ext3, illetve ReiserFS használatára. Fontos dolog a naplózás, én az ext3-at javaslom. Mivel ez egy egyszerű ext2-es rendszer egy apró naplóállománnyal kiegészítve, a rendszert az előző, csak az ext2-t ismerő rendszermagokkal is életre lehet kelteni, legfeljebb nem lesz naplózás. Az egyre nagyobb merevlemezek terjedésével talán keveseket érint a gond, de lényeges, hogy a ReiserFS 32 MB-os naplóméretével szemben az ext3 8 MB-ot foglal el. Egy kissé szokatlan volt még a ***Kezelőprogramok (modulok) konfigurálása*** menüpont. Az eddigi logikai csoportosítás helyett a modulok fizikai elhelyezkedésük szerint lettek csoportosítva, vagyis ahogyan a ***/lib/modules/2.4.18/kernel*** könyvtár alatt szerepelnek, ahhoz hasonlóan jelennek meg itt is.

Újraindítás után a megszokott beállítások következnek: időzóna, árnyékjelszavak, rendszergazdai jelszó. Végre működik a `tasksel`, ami nagyon fontos a Debiannal most ismerkedők számára. Itt témák szerint összeválogatott csomagcsokrokat lehet egyszerűen és gyorsan telepíteni. Legvégül elindul a `dselect`. Ajánlom a következő csomagok telepítését, mert igen megkönnyítik mindennapi munkánkat:

mc – Midnight Commander
mc-common – a Midnight Commanderhez szükséges
libgpmg1 – gpm-függvénykönyvtár (libc6)
bzip2– tömörítő, kicsomagoló
libbz2-1.0 – bzip2-függvénykönyvtár
aptitude – új felület az apt-hez, nagyon barátságos
discover – felismeri az eszközöket és betölti a hozzá tartozó modulokat
discover-data – adatállományok a discoverhez
libdiscover1 – discover-függvénykönyvtár
ash – NetBSD-héj, a discover igényli
bsdgames – a legfontosabb (bsd-játékok)

Továbbá érdemes telepíteni egy a processzorodhoz megfelelő rendszermagot. Ha nem is szeretnél a rendszermag-újrafordítással bíbelődni, egy a processzorhoz előfordított rendszermaggal az egész rendszer sokkal gyorsabb lesz. Csupán ki kell választani a `kernel-image-verzio-cpu` csomagot, és telepíteni kell. Ezután a LILO-t is be kell állítani. A 2.4-es sorozat képviselői közül elérhető a 2.4.16 és a 2.4.18, a következő változatokban:

386 – 386-os processzorhoz
586 – 586/K5/5x86/6x86/6x86MX-os processzorhoz
586tsc – eredeti Pentium I processzorhoz
686 – Pentium Pro/Celeron/Pentium II/Pentium III processzorhoz
686smp – az előző többprocesszoros változata
k6 – AMD K6/K6-II/K6-III processzorhoz
k7 – AMD K7 processzorhoz

Vélemény

Elég régóta Debian-hívő vagyok, és úgy gondolom, a Woody az utolsó pillanatban érkezett. A Potatóban már annyira elavultak a csomagok, hogy éles rendszereken is Woodyt használtam, annak ellenére, hogy még nem volt megbízható és kipróbálás alatt állt. A Debian jó rendszer, mert kicsi az alapcsomag, amit feltétlenül telepíteni kell, és nagyon jó a csomagkezelője. Megbízható, ennek ugyanakkor az az ára, hogy lassan fejlesztik. Megérte a két év várakozást? Egyértelműen igen.

Nem szabad azonban elhallgatni a hiányosságokat sem. A telepítés alatt `debconf` -fal beállított `locales` csomag ***/etc/locale.gen*** beállításfájlja a telepítés végére üres, akárhány `locale` -t állítok is be, és nem hozza létre a megfelelő állományokat. Utólag újra be kell állítani, ami telepítés közben nem sikerült:

```
# dpkg-reconfigure locales
```

Még mindig nem értem, hogy miért az `exim` az alapértelmezett MTA (Mail Transfer Agent – levéltovábbító ügynök). Miért nem lehet kiválasztani, hogy mit akarok használni? Én Postfix-párti vagyok, és minden Debian-telepítés után az első, hogy lecserélem az eximet.

Végsősoron amire vártunk, megérkezett. Egy frissebb Debian 2.4-es rendszermaggal, `IP Tables` -támogatással, `pppoeconf` -fal az ADSL beállításához, PGP-vel és ext3-as fájlrendszerrel. Még több felirat lett magyar nyelvű, és az `aptitude` révén a `dselect` miatt elriadt felhasználók is talán adnak neki még egy esélyt. Nagy munkát végeztek a srácok, úgyhogy egy próbát mindenkinek megér. Sok szerencsét hozzá!

Fülöp Balázs
(xut@freemail.hu) 17 éves, imádja a Túró Rudit, a Debian Linuxot és a teheneket. Az ELTE Radnóti Miklós Gyakorlóiskola tanulója immár ötödik éve. Kedvenc írója Slawomir Mrożek. Leginkább a számítógépes hálózatok biztonsága érdekli.

Naprakésznek maradni az épelméjűség határain belül

Hogyan segítenek a SuSE YaST2, a Red Hat up2date és a Debian apt-get eszközök rendet tartani a biztonsági foltok erdejében?

A biztonság mozgó célpont, könnyen elszédül az ember, ha követni próbálja. A Linuxhoz hasonló bonyolult rendszereken az egyik legunalmasabb és legfárasztóbb biztonsági feladat az alkalmazások, a parancsok és a programkönyvtárak naprakész követése. Bárki tanúsíthatja, aki olvassa a Bugtraq-listákat, hogy állandóan kihasználható programhibákat fedeznek fel, és javítófoltokat adnak közre. Egy átlagos Linux-terjesztés többszáz csomagját tekintve senki sem remélheti, hogy lépést tud velük tartani.
Ez nem hangzik túl jól, de sajnos igaz. Mégha módunkban is áll figyelemmel kísérni az összes létező biztonsággal foglalkozó levelezőlistát, és minden egyes közzétett sebezhetőségi pontot befoltozunk, előbb-utóbb mi magunk is egy hibajelentés főszereplőjévé válhatunk. Bizonyos hibák addig nem kerülnek napvilágra, amíg valaki ki nem használja őket.
Vigasztalásul elmondhatjuk, hogy ha felzárkózni nem is lehet teljesen, minden előrehaladás sikerként könyvelhető el. Ha egy háromhetes programhiba miatt törik fel a rendszerünket, még mindig menőbbek vagyunk, annál, akit egy hároméves hibával találnak meg (másrészről akárhogy is nézzük, a betörés az betörés).
Félretéve a tréfát: tisztán statisztikai alapon is belátható, hogyha kevesebb foltozatlan hiba van, kevesebb a sebezhető pont. Annak ellenére, hogy mennyire hálátlan és végeláthatatlan feladatnak tűnik, érdemes megpróbálkozni Linux-telepítésünk naprakészen tartásával. Szerencsére a népszerű Linux-terjesztések újabb változatai tartalmaznak olyan programokat, amelyek a feladat nagy részét elvégzik helyettünk. Ilyenek a SuSE-ben a YaST2, a Red Hatben az `up2date` és a Debianban az `apt-get` (ezek közül egyesek még biztonságosak is!).

Csomag vagy forrás?

A legfontosabb tanáccsal kezdeném, amire sokévnyi kételkedés után jöttem rá: amikor csak lehet, használd a terjesztésed által támogatott csomagot. Ez sok ember számára talán természetesnek hangzik – miért fordítanánk le valami forrásból, ha nem muszáj? A régi motorosok viszont mindent forrásból fordítanak, számukra ez a megszokott, és ehhez értenek. Ez azért alakult így, mert az 1990-es évek elején sokkal kevesebb program volt Linux alatt, mint ma, így sok mindent olyan forrásból kellett fordítani, amit eredetileg más felületekre fejlesztettek. Akkoriban a Linux terjesztésekbe való csomagolásának módszerei sem voltak még annyira fejlettek, kevesebb volt a terjesztés is, és közel sem változtak ilyen gyorsan.
(Kellett-e már valaha forrásból fordítani a `ps` parancsot, mert vadonatúj rendszermagod nem képes együttműködni a terjesztéseddel együtt érkezett régebbi `ps`-változattal? Néhányunk átélte ezt a felemelő élményt. Ezenkívül minden nap 15 kilométert kellett gyalogolnunk az iskoláig kígyóktól hemzsegő mocsarakon át stb.)
Ma már másként mennek a dolgok. Ne érts félre, nem mondom, hogy legyünk a bináris csomagok rabjai. Megeshet, hogy egy alkalmazás olyan tulajdonságára van szükséged, amely hiányzik a terjesztésben szereplő változatból, vagy a saját ízlésed szerint szeretnéd lefordítani a programot, mert nem felel meg a terjesztésben szereplő egyencsomag. Ennek ellenére az esetek többségében sok fontos előnye van a bináris csomagok használatának.
Az első a kényelem: sok alkalmazás egyetlen helyről való letöltése gyorsabb és könnyebb, mint ha egyesével kellene leszedegetni őket a fejlesztők honlapjairól (főként ezért alakultak ki a terjesztések), és a bináris csomagok telepítése sokkal gyorsabb és hibátlanabb, mint a fordítás. A kényelem nem az az érték, amit mi, vakbuzgó unixosok különösebben elismerünk, de azért el sem hanyagolandó.
A második előny a csomagok megbízhatósága, ugyanis a nagyobb terjesztéseknél a csomagolás előtt az alkalmazásról eldöntik, hogy érdemes-e betenni a terjesztésbe, és ha igen, melyik a legmegbízhatóbb változata, milyen fordítási beállítások illenek legjobban a terjesztéshez stb. Volt már ez alól egy pár híressé vált kivétel, manapság azonban a legtöbb terjesztésnél elég rendesen ellenőrzik a termék minőségét.
A megbízhatóság magától értetődően a biztonság szempontjából is fontos: ahol hibák vannak, ott sebezhető pontok is akadnak. Még az olyan hibákat is ki lehet használni, amelyeknek nincs nyilvánvaló biztonsági következményük, gondoljunk csak a szolgáltatásmegtagadásos (DoS-típusú) támadásokra, amelyeknek az a célja, hogy lefagyasszák a rendszert.
Ez elvezet az alkalmazások biztonságának egyik kiábrándító ellentmondásához: bár az Interneten leginkább elterjedt sebezhetőségek a gyengén karbantartott alkalmazásokból erednek (elavult, illetve ismert hibával bíró változatok), az újabbak nem feltétlenül jobbak. A biztonsággal foglalkozó szakemberek az évek során joggal marasztalták el a Microsoftot, hogy csigalassúsággal ismeri el a hibáit, és termékeihez ráérősen adja ki a javításokat. De ugyanúgy keserű tiltakozás követi azt is, amikor a Microsoft gyorsan ad ki egy a megbízhatóságot rontó javítófoltot, mert ez nagymértékben csökkenti az eredeti hiba kijavításával elért előnyt.
Most egy pillanatra hagyjuk figyelmen kívül, hogy a megbízhatóság önmagában is kívánatos, és gondolkodjunk el a következőkön: tegyük fel, hogy egy alkalmazásban lehetőség kínálkozik egy jelentéktelen tártúlcsordulási hiba elméleti kihasználására a rendszergazdai jogok szerzésének céljából, de ezt csak olyan felkészült assembly-programozók tudják kiaknázni, akik az RC4-folyamok titkosításához is jól értenek. Ezt egy olyan kódrészlettel foltozzuk meg, ami miatt lehetőség nyílik a gép lefagyasztására, amit még a jobb elfoglaltságot nem találó iskolás kölykök is ki tudnak használni. Nagyobb lett ettől a biztonság? A válasz a pontos körülményektől függ és attól, hogy kinek tesszük fel a kérdést, de a tanulság az, hogy a programfrissítések gyakran ilyen gondokat vetnek fel.
Talán túlmagyarázom ezt a témakört, de fontos, hogy eloszlassuk azt a tévhitet, miszerint az azonnali programfrissítés valamiféle csodaszer. Évekig tartott, míg teljesen megértettem, hogy például miért csomagolják a legeslegújabb OpenBSD-

terjesztésbe is a BIND v.4.9.8-at, ami számítógépévekben mérve már ősöregnek számít.
Azért is fontos ez – eredeti témánkhoz visszatérve –, hogy megértsük, miért célszerű ráhagynunk Linux-terjesztésünk összeállítóira a nehéz foltozási döntések meghozatalát, és a foltozással a terjesztés által kiadott hivatalos (és remélhetőleg kipróbált) javítást megvárnunk. Épp elég nehéz lépést tartani a terjesztés által kiadott biztonsági frissítésekkel – teljesen kilátástalannak tartom, hogy saját magam foltozzam és fordítsam újra az összes lényeges alkalmazást, és izgatottan várjam, hogy a foltozással nem rontottam-e el valami mást is.
Összegezve az eddigi bölcsességeket: naprakésznek lenni erény, bináris csomagokat használni a lustaság erényes formája, és a másodkézből származó (a Linux-terjesztésed által kiadott) biztonsági frissítések használata az egyéni „vagánykodás" helyett nem lustaság, hanem okosság.

Melyik rendszert kell frissíteni és hogyan?

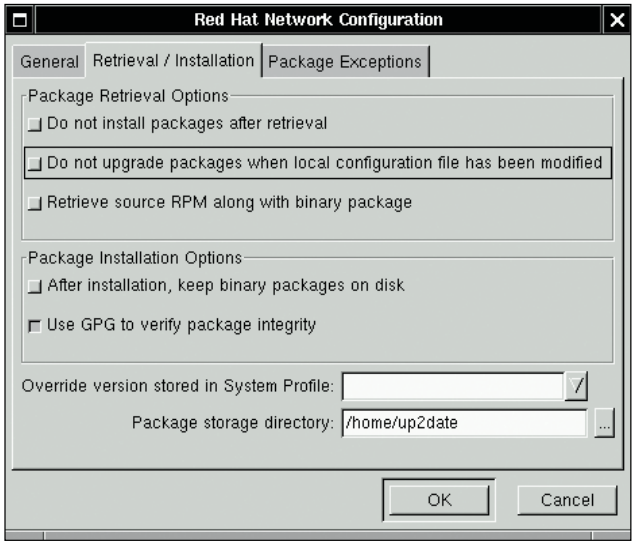
Mielőtt az önműködő frissítési eszközökről szólnék, beszéljünk egy kicsit a rendszerek különféle fajtáiról. Az Internetre közvetlenül csatlakoztatott kiszolgálónak sokkal sürgetőbb igénye van az azonnali alkalmazásfrissítésre, mint egy tűzfal mögött elhelyezett asztali gépnek. A józan ész segít annak a döntésnek a meghozatalában, hogy melyik rendszernél kell több időt fordítani a szigorú naprakészen tartásra, és melynél fogadható el alacsony kockázati szint a kisebb energiaráfordítás következtében.
A másik dolog, amit a rendszerek szerepe kapcsán meg kell fontolni: melyik gépen futtatható kényelmes X-alapú frissítőprogram, mint amilyen például a YaST2? Az Internetre kapcsolt kiszolgálón nem tanácsos X-kiszolgálót futtatni. Ha nagyon tetszik a YaST2 könnyű használhatósága és az általa biztosított kényelem, futtasd egy belső rendszeren, állítsd be a frissítések mentését, és a frissítéseket `scp`-vel másold fel a külső gépre.

Red Hat up2date

A Red Hat 6.2 óta az `up2date` program használható a frissített csomagok önműködő azonosítására, letöltésére és telepítésére. Az `up2date` szöveges és grafikus módban is futtatható, így a „Hol futtathatom?" fent taglalt kérdése fel sem merül nála.
Az `up2date`-et a futtatás előtt be kell állítani. Szerencsére két egyszerű és kényelmes eszköz is a segítségünkre van. Először létre kell hozni egy felhasználói fiókot a Red Hat Networkön (RHN) az `rhn_register` használatával. Kapcsolók nélkül indítva az `rhn_register` grafikus módban indul, de a `nox` kapcsolót megadva szöveges módban is futtatható::

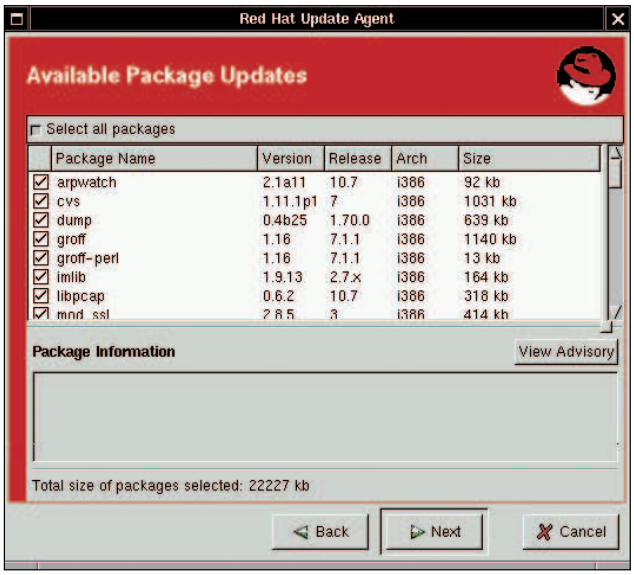
```
bash-# rhn_register -nox
```

Az RHN-nél egy rendszer díjmentesen bejegyeztethető, azonban az ugyanahhoz a fiókhoz tartozó további rendszerek után díjat kell fizetni. RHN-bejegyzés szükséges minden olyan gépre, amelyen az `up2date`-et futtatni akarod, viszont semmi sem akadályoz meg benne, hogy egyetlen gépen futtasd az `up2date`-et, majd mentsd a frissített csomagokat, és többi Red Hat rendszeredre `scp`-vel vagy más biztonságos módon átmásold, majd a rendszereken kézzel telepítsd őket.
Egy további dolog, amit tudnod kell az RHN-ről, hogy az RHN-be bejegyzett rendszer alapértelmezés szerint a számítógép hálózati beállításait, alkatrészeinek listáját és az összes telepített csomag nevét, valamint változatszámát elküldi az RHN-nek. Ennek segítségével ütemezheted az RHN által küldött önműködő frissítéseket, illetve testreszabott leveleket kaphatsz, ha a gépedre telepített valamelyik csomaghoz frissítés jelent meg.
Valószínű, hogy ez a két lehetőség tetszeni fog neked, és amennyiben a kettő közül bármelyik is megkönnyíti a rendszeres biztonsági frissítések kezelésével kapcsolatos munkádat, nem foglak bírálni érte. Nekem azonban nincs ínyemre, hogy idegeneknek részletes listát adjak a gépem beállításairól. Nem arról van szó, hogy okom lenne a bizalmatlanságra a Red Hatnél dolgozó remek szakemberekkel szemben, csak azt állítom, hogy nem kell megbíznom bennük, ha nincs kedvem hozzá.
Végül is nem olyan nagy feladat a Redhat-Watch listát járatni, amelyen a Red Hat bejelenti a frissítéseket (➲ http://listman.redhat.com), és ezután eldönteni, hogy futtatni kívánom-e az `up2date`-et. Ha szükséges, az `up2date` akkor is megállapítja, hogy mely frissítések érintik a rendszeremet, ha a Red Hat semmilyen adatot nem tárol rólam. Emiatt én azt a gyakorlatot követem (és mindenkinek ezt javaslom), hogy ne jelöljük be az `rhn_register` következő beállításait: ***Include information about hardware and network*** (Az alkatrészek és a hálózat adatainak továbbítása) és ***Include RPM packages installed on this system in my System Profile*** (Telepített RPM-csomagok feljegyzése a felhasználó rendszerprofiljában).

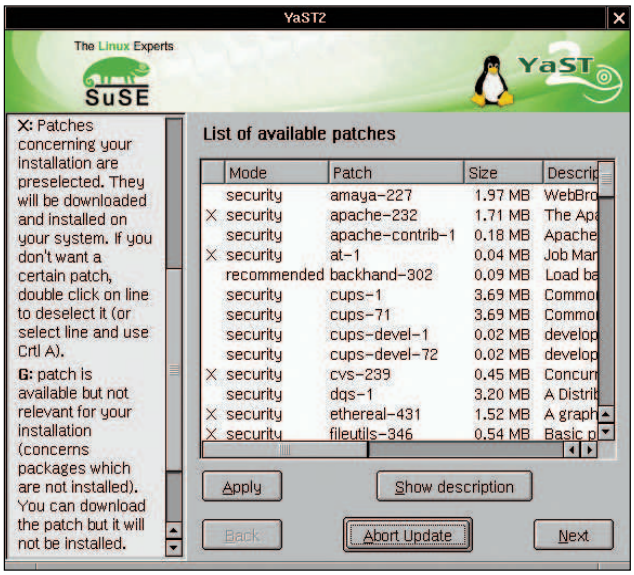


1. kép A Red Hat up2date-config eszköze

Miután rendszeredet bejegyeztetted a Red Hat Networknél, futtatnod kell az `up2date-config` programot (lásd az *1. képet*). Az `rhn_register`-hez hasonlóan ez a parancs is támogatja a `nox` kapcsolót, ha szöveges módban kívánod futtatni. Az `up2date-config` használata többnyire magától értetődik, de egynéhány beállítást érdemes megemlíteni.
Először is: ha a fent leírt módon egy központi gépet szeretnél használni a frissítések tárolására és terjesztésére, az ***After installation, keep binary packages on disk*** (Telepítés után a binárist hagyja a lemezen) beállítást be kell jelölnöd, és a ***Package storage directory:*** (A csomagok tárolási helye) alapértelmezett értékét is meg kell adnod (vagy legalább megjegyezned). Ha az `rhn_register`-t a `-nox` kapcsolóval futtattad, ezek a beállítások ***keepAfterInstall*** és ***storageDir*** néven szerepelnek.



2. kép Az up2date egyik képernyője



3. kép Frissítés a YaST2-vel

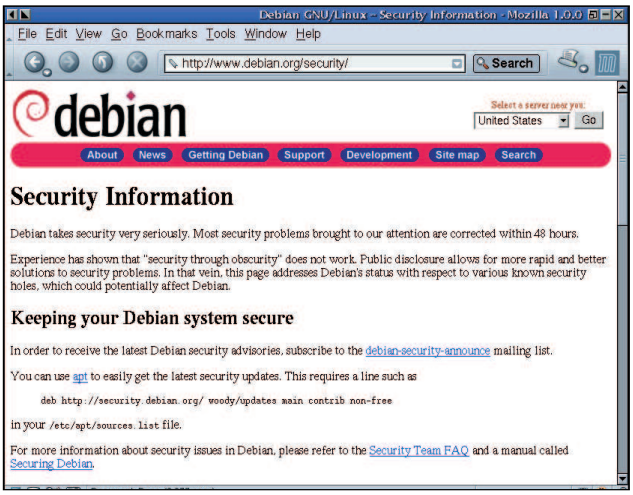
Másodszor: hacsak nem túl lassú a géped, ne tiltsd le az alapértelmezés szerint engedélyezett ***Use GPG to verify package integrity*** (GPG használata a csomag épségének ellenőrzésére) beállítást. Az rpm formátum egyik legjobb tulajdonsága a belső GPG-aláírás használata, amellyel ellenőrizhető a csomag épsége. Az `up2date` ezt az ellenőrzést magától elvégzi, ha a GnuPG telepítve van a gépen. Egyénként az RPM-csomagok GPG-aláírását a következő paranccsal kézzel is ellenőrizheted:

```
rpm --checksig /path/packagefilename.rpm
```

Természetesen a */path/packagefilename.rpm* helyére az ellenőrzendő RPM-fájlt kell írni, a teljes elérési úttal.
Előbb vagy utóbb valaki feltöri a Red Hat egyik tükörkiszolgálóját, és valamelyik létfontosságú csomagot egy trójai falóval helyettesíti. Ha ez megtörténik, akkor azokat, akik rendszeresen ellenőrzik az RPM-csomagok aláírását, ez a gond sokkal kevésbé fogja érinteni.

Milyen gyakran frissítsd az egész terjesztést?

Tegyük fel, hogy a képzeletbeli Bo-Weevil Linux 33.1 terjesztést használod, és mindig elvégezted a szükséges csomagfrissítéseket. Mi a teendő, ha megjelenik a Bo-Weevil 33.2? A folyamatos naprakész állapot fenntartásához szükséges-e a 33.2-re történő frissítés?
Lehetséges, de valószínűleg nem. A terjesztés frissítése (a csomagfrissítésekkel szemben) általában új tulajdonságokat, csomagokat és igen, biztonsági frissítéseket is ad a rendszerhez, azonban a terjesztés frissítése önmagában nem biztosítja, hogy a rendszer biztonságosabbá válik. Más szavakkal kifejezve: a teljesen megfoltozott Bo-Weevil 33.1 valószínűleg biztonságosabb, mint a foltozatlan 33.2, és legalább olyan biztonságos, mint a foltozott 33.2-es rendszer.
Csak akkor frissítsd a terjesztést, ha olyan új tulajdonságokkal rendelkezik, amelyekre szükséged van. Azonnal frissíts, ha az általad használt változat támogatása hivatalosan megszűnik (például több biztonsági frissítést nem adnak ki). Ha egyik eset sem áll fenn, továbbra is frissítgesd a régebbi terjesztés csomagjait, és ne ess a változatszámhóbort hibájába.

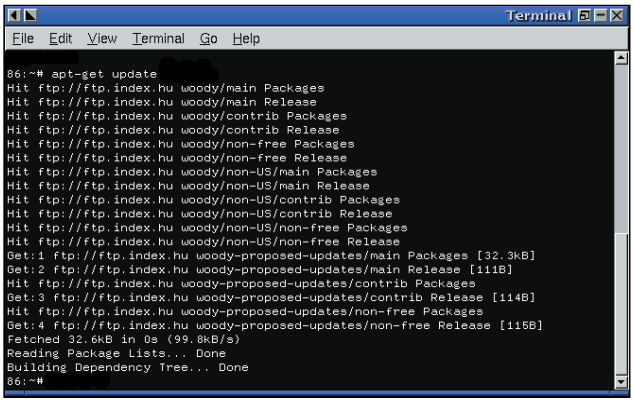


4. kép A Debian/GNU biztonsággal foglalkozó weboldala

Miután bejegyeztetted magad az RHN-nél, és lefuttattad az `up2date-config`-ot, futtathatod magát az `up2date` programot (lásd a *2. képet*). Erről nem lehet túl sokat mondani, az `up2date` célja eleve az egyszerűség és a kényelem, szükségtelen ecsetelnem, hogyan kell a világosan feliratozott gombokra kattintgatni.
Megismétlem korábbi tanácsomat: iratkozz fel a Redhat-Watch listára, és amikor a rendszeredet érintő frissítést bejelentik, azonnal futtasd az `up2date`-et. Egy ilyen remek és felhasználóbarát eszköz birtokában nincs mentséged, ha elmulasztod.

A SuSE frissítőrendszere

Ha SuSE Linuxot használsz, még egyszerűbb a dolgod. Az önműködő frissítésekhez a YaST2 frissítőmodulját használhatod (lásd a *3. képet*).



5. kép Az apt-get update parancs kimenete

A Red Hat `up2date`-tel ellentétben a SuSE-nél a számítógépet nem kell bejegyeztetni a program futtatása előtt, és külön alkalmazás vagy fájl segítségével beállítani sem szükséges.
A frissítőmodul mindent egyben tartalmaz. Az első néhány képernyőjén szükség esetén megváltoztathatóak a beállítások (például a csomagok letöltéséhez a tartózkodási helyedhez közelebbi kiszolgálót is választhatsz, mint például a SuSE amerikai ➲ ftp.suse.com kiszolgálója).
Ha tudni akarod, mikor kell elindítani a frissítőmodult, iratkozz fel a SuSE suse-security-announce levelezőlistára. A Redhat-Watch listához hasonlóan a forgalom kicsi, és ne aggódj amiatt, hogy a SuSE komolytalan üzenetekkel szemeteli tele a postaládádat. A feliratkozáshoz látogass el a ➲ http://www.suse.com/en/support/mailinglists/index.html címre.
Őszintén szólva sokkal többet nem kell elmondanom a YaST2 frissítőmoduljáról, kivéve egy apró hibát, amit tapasztaltam. Tulajdonképpen a felhasználó hibája, de könnyű elkövetni.
Ha a `yast2` parancsot egy `xterm`-ből, vagy a ***Parancs futtatása*** párbeszédablakon keresztül kézzel adod ki, és az X nem rendszergazdaként fut, a frissítés sikertelen lesz, és azt a félrevezető hibaüzenetet kapod, hogy a frissítési lista a megadott FTP-kiszolgálón nem érhető el.
Nem ez az igazi ok, valójában a YaST2 rendszergazdai jogosultságokat igényel ahhoz, hogy ezt a fájlt kiírja a lemezre, miután az FTP-kiszolgálóról letöltötte. Ez nem azt jelenti, hogy a YaST2csak a rendszergazda által futtatott X-munkafolyamatból indítható, de ha az X nem így fut, akkor a YaST2-t a SuSE által létrehozott menübejegyzés segítségével kell elindítani, mert ebben az esetben elkéri a rendszergazda jelszavát, és a frissítés rendben megtörténik.

Az RPM-csomagok kézzel történő frissítése

Annyira fellelkesített az `up2date` és a YaST2, hogy még nem is említettem az RPM-fájlok frissítésének egyik legegyszerűbb módját: magát az `rpm` parancsot. Ez a módszer ugyanolyan jól működik a Red Hat (és származékai), valamint a SuSE alatt. Használatát egy példán keresztül mutatom be.
Tegyük fel, hogy értesítést kaptál a képzeletbeli SuSE vagy Red Hat „blorpflap” csomag sebezhetőségéről és az azt megszüntető frissítésről. A frissített RPM-csomagot az értesítésben megadott címről letöltötted a helyi ***/usr/pkg/updates/blorpflap-3.2-3.rpm*** útvonalra. Először ellenőrizd a csomag érvényességét:

```
rpm --checksig /usr/pkg/updates/
➥blorpflap-3.2-3.rpm
```

Természetesen ehhez az szükséges, hogy a terjesztés csomagaláíráshoz használt kulcsa rajta legyen a nyilvános GnuPG kulcscsomódon (a Linuxvilág 2001. októberi és novemberi számában megjelent egy kétrészes írásom a GnuPG használatáról).
Miután a GPG-aláírás ellenőrzése rendben zajlott (vagy feltételezed, hogy rendben zajlana le, amit a saját felelősségedre megtehetsz – a fenti lépés nem kötelező), telepítsd a frissítést:

```
rpm -Uvh /usr/pkg/updates/blorpflap-3.2-3.rpm
```

A `-U` természetesen az update (frissítés) rövidítése (valójában frissítés vagy telepítés), és már telepített csomagok frissítésére vagy új csomagok telepítésére használható; a `-v` bőbeszédűvé teszi a folyamatot; a `-h` hatására pedig a folyamat előrehaladásáról képet kaphatunk.

A Debian apt-get programja

Ebben a hónapban ez az utolsó eszköz, amit bemutatok.
A Debian általános jellegzetességének megfelelően az `apt-get` kevésbé csillog-villog, csakhogy bizonyos szempontból könnyebben használható, mint a többi terjesztés díszes grafikus frissítőeszközei. Dióhéjban elmondva: a `deb`-csomagok frissítése a Debian rendszeren két lépésből áll:

1. a csomaglista frissítése,
2. az új csomagok letöltése és telepítése.

Mindkét lépés az `apt-get` segítségével hajtható végre:

```
bash-# apt-get update
bash-# apt-get -u upgrade
```

A második parancs hatására az `apt-get` a `wget` segítségével letölti a frissített csomagokat és telepíti őket.
A Debian biztonsági hibáiról és a frissítésekről értesítést kaphatsz, ha feliratkozol a debian-security-announce levelezőlistára a ➲ http://www.debian.org/MailingLists/subscribe oldalon található űrlap kitöltésével. Futtasd le az `apt-get` programot, ha értesítést kapsz egy a rendszeredet érintő biztonsági frissítésről.
Bármennyire is szeretem az `apt-get` programot, akad egy fontos hiányossága: nem lehet vele GPG-aláírásokat ellenőrizni. Ez azért van így, mert a deb formátum sajnos nem támogatja az aláírásokat, és a Debian-csomagokat jelenleg külső aláírással sem látják el. Állítólag a deb formátum következő változata már támogatni fogja a GPG-aláírásokat.

Ennyi jutott erre a hónapra. Sok szerencsét kívánok a frissítésekhez!

Linux Journal 2002. július, 99. szám

Mick Bauer
(mick@visi.com) hálózati biztonsági tanácsadó az Upstream Solutions Inc.-nél Minneapolisban (Minnesota). Mick a szerzője a hamarosan megjelenő új O'Reilly-könyvnek, amelynek címe „Building Secure Servers With Linux”, de ő írta a „Network Engineering Polka” című művet is. Büszke apja gyermekeinek.

Grafikus felület programozása Qt-ban (1. rész)

Írásunkkal a grafikus felhasználói felülettel rendelkező programok készítésének rejtelmeibe kívánunk betekintést nyújtani. Készítsünk egyszerű szövegszerkesztő programot együtt! A cikk második részében bővíteni fogjuk a program képességeit, valamint elkezdjük a honosítását is

Már régen elmúltak azok a napok, amikor a grafikus felhasználói felület készítése egyet jelentett kedvenc karakteres szerkesztőnk és a `make` órákon keresztüli nyuvasztásával. Ma már a bonyolult fejlesztői felülettel (amilyen például a KDevelop) nem rendelkező fejlesztői rendszerek is a súgójukban azt ígérik, hogy a grafikus felületek fejlesztését a lehető legfájdalommentesebbé teszik.

Aki azt hiszi, hogy a KDE által is használt Qt fejlesztőcsomag mindössze egyetlen programkönyvtárból áll, nagyon meglepődik, amikor egy 14 MB-ot meghaladó méretű tárállományt tölt le a ➲ http://ftp.trolltech.com/pub/qt/source címről. Annak ellenére, hogy a Qt korábbi változatai híresek voltak megbízhatóságukról, a Qt 3.0-nak 2001. október közepi első megjelenése óta annyi hibája vált ismertté, hogy már a negyedik javítócsomag jelent meg. Sőt, további csomagok megjelenése várható. Emiatt minden esetben a legfrissebb csomag használatát javasoljuk.

A `qt-x11-free-3.0.x.tar.gz` tárállományban – mi magunk a Qt 3.0.4-et használtuk – nem csupán az osztálykönyvtárat találjuk meg, hanem annak HTML formátumú leírását is, valamint több olyan eszközt, amelyek életünk megkönnyítését ígérik a grafikus alkalmazásfejlesztés területén.

Egy egyszerű szövegszerkesztő-alkalmazás grafikus felhasználói felületének elkészítéséhez szerezzük be a Qt Designer programot. Az említett szövegszerkesztő megtalálható a 40. CD Magazin/Qt könyvtárában. A programmal együtt birtokunkba kerül egy felhasználóifelület-fordító eszköz (User Interface Compiler – UIC), amely a Designer program XML formátumú kimenetét C++ nyelvű programmá alakítja. Írásunk második részében további C++-kóddal egészítjük ki kedvenc szövegszerkesztőnket, hogy életet vigyünk grafikus felületébe.

A jövő hónapban kitérünk a Qt fejlesztőkörnyezet újabb tagjára, a Qt Linguistre is, mellyel elkezdjük a program honosítását. A Qt Designerhez hasonlóan ez a segédeszköz is grafikus felülettel rendelkezik, valamint XML formátumot használ ki- és bemenetként. Az XML formátumú fordítást igénylő kifejezéslista elkészítéséhez a programcsomagban egy parancssori eszközt kapunk, amit az `lupdate` névre kereszteltek. Amint a fordítás elkészült, egy másik parancssori eszköz, az `lrelease` az XML-állományt az alkalmazás futása idején a szükséges bináris formátumúvá alakítja.

`Makefile`-okat készíteni Qt-alkalmazásokhoz távolról sem gyerekjáték. A Qt régebbi változatai nem tartalmaztak ehhez a feladathoz segédeszközt; a Qt gyártója, a Trolltech a `tmake` nevű eszközt kínálta külön letöltésre. A Qt 3.0-s tárállományban szerepel a `qmake` nevű új segédprogram, ami nagy segítségünkre lehet a `Makefile` létrehozásához. Erre is a cikk második részében térünk ki.

Tervek szövögetése

A g++ fordítókörnyezettel az összes szükséges fejlesztőeszköz a birtokunkban van; most már csak az elkészítendő szövegszerkesztő-alkalmazásra kell összpontosítanunk. De vajon milyen igényeket is támasszunk programunkkal szemben?

Mivel egyszerű szövegszerkesztőt szeretnénk csak készíteni, csupán az alábbi feladatokat várjuk el: új szerkesztőablak megnyitása és bezárása, dokumentum mentése más néven és kilépés az alkalmazásból. Természetesen tudjon szöveget másolni, kivágni, beilleszteni. Szeretnénk a visszavonás-műveletet is elérni, valamint megköveteljük a programtól, hogy a betűjellemzőknél a dőlt, félkövér és aláhúzott formázást lehessen alkalmazni, illetve ezek tetszőleges kombinációit is. A szerkesztő ***Névjegy*** ablakában jelenjen meg pár sor, a neve pedig legyen mondjuk, `ljedit`.

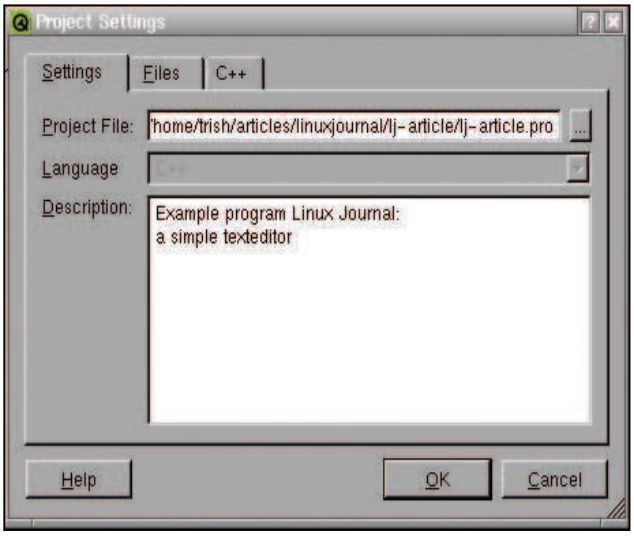
A betűjellemzőktől eltekintve minden feladat legyen elérhető a ***Fájl***, ***Szerkesztés*** és Súgó menün keresztül. A dőlt, félkövér és aláhúzott betűtulajdonságok kapcsolói az eszköztár almenüiként legyenek hozzáférhetők, és ugyanitt további ikonok szerepeljenek az állományok megnyitásához és mentéséhez, a műveletek visszavonásához és megismétléséhez, a szövegelemek kivágásához, másolásához és beillesztéséhez.

Minden ikonhoz tartozzon előugró súgó (buborék), ami jelenjen meg, ha a felhasználó az egérrel elidőz az egyes ikonok felett. A ***Mentés másként*** és a ***Kilépés*** szolgáltatásokat kivéve valamennyi lehetőség gyorsbillentyűkön keresztül is legyen elérhető.

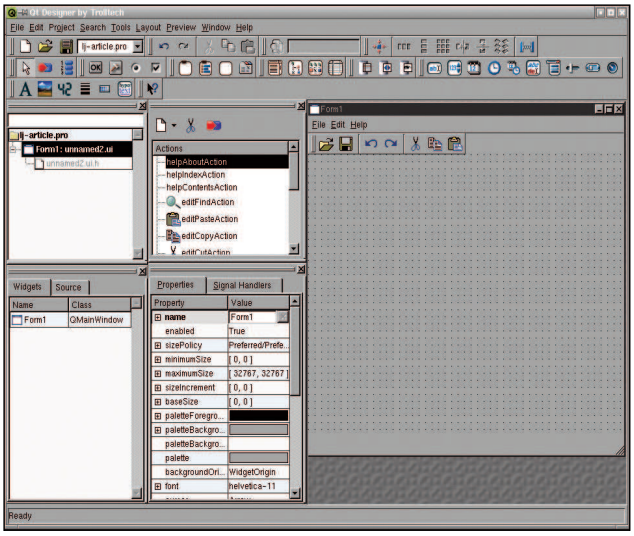
A felhasználókat a váratlan adatvesztéstől megóvandó az állományok megnyitásakor és bezárásakor, továbbá az alkalmazásból történő kilépéskor jelenjen meg a kérdés, hogy a felhasználó a régi állomány (megváltozott) tartalmát menteni szeretné-e, vagy veszni hagyja a módosításokat. A fentebb említett szolgáltatások többségének tervezése a Designerrel elvégezhető, az utóbbi feladat megvalósítására azonban ugyancsak következő számunkban térünk ki.

A felhasználói felület megtervezése

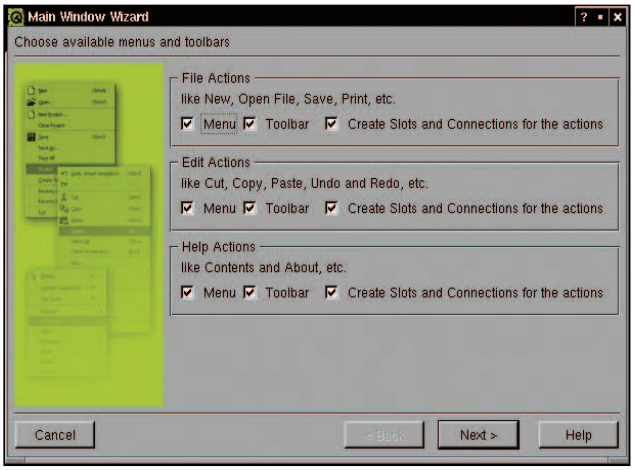
Amennyiben a rendszerre egynél több Qt rendszer van telepítve, a `QTDIR` változót a megfelelő Qt-változatot tartalmazó könyvtárra kell irányítani. Ezután egy terminálból a `designer &` paranccsal indítsuk el a fejlesztőkörnyezetet. Ha a `$QTDIR/bin` könyvtár nincs a keresési útvonalban, vagy több Qt-változat is telepítve van, a parancs kiadásánál érdemes használni a teljes útvonalat. Új feladatfájlt a ***File*** menü ***New*** pontjával hozhatunk létre. A megjelenő párbeszédablakban kattintsunk a ***C++ Project*** ikonon, majd döntésünk megerősítése végett kattintsunk az ***OK*** gombra. Most már létre tudunk hozni új qmake-feladatot, ha megadjuk a nevét, a helyét és



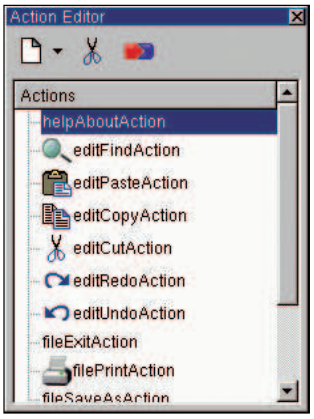
1. kép Az első lépés: a projekt létrehozása



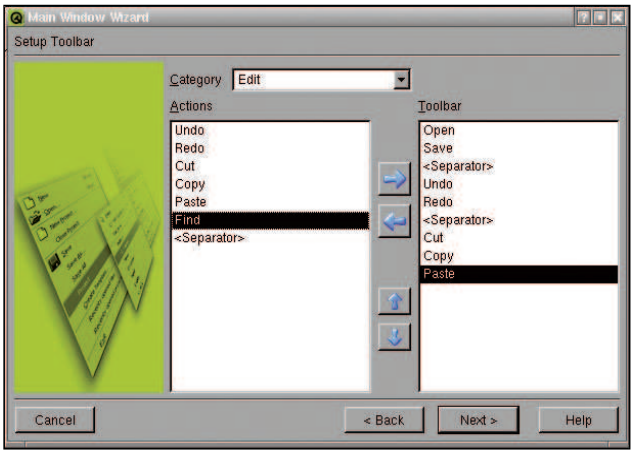
4. kép A Designer és az új form



2. kép A menük és az eszköztár létrehozása

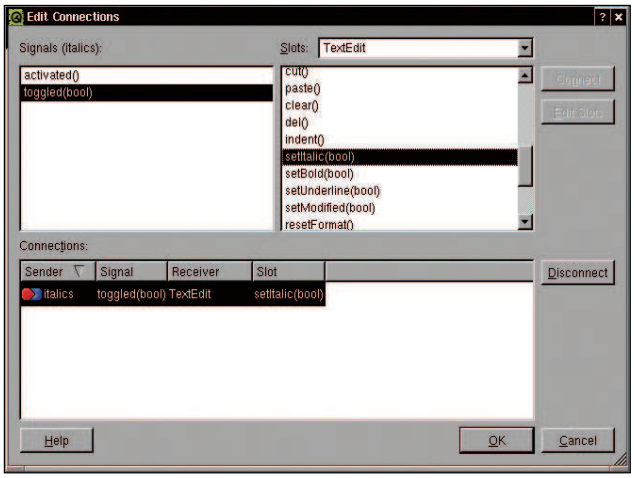


5. kép Az Action Editor

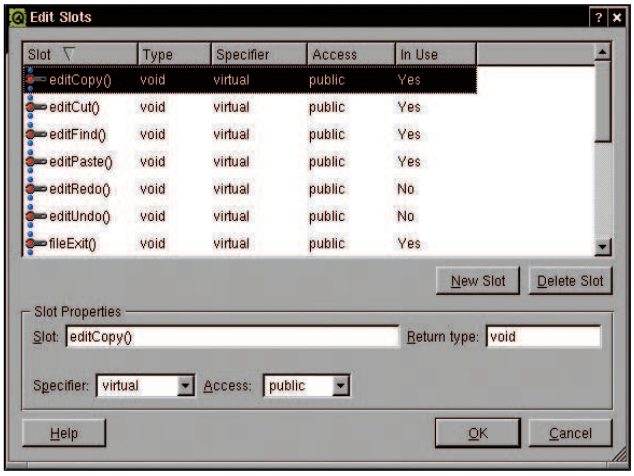


3. kép Töltsük fel az eszköztárat!

a leírását (lásd az 1. *képet*).
Ezután még egyszer a ***Fájl*** menüben válasszuk a ***New*** pontot, hogy létrehozzuk programunk első (és egyben a cikkben felsorolt igények számára szükséges egyetlen) grafikus elemét (widget). Válasszuk tehát a ***Main Window*** ikont, létrehozva a program főablakát.
Ha telepítve van a varázsló, akkor most megjelenik a Main ***Window Wizard*** ablak első oldala (lásd a 2. *képet*). Egy pillanatig elgondolkodunk, majd igen, gondoljuk úgy mi is, hogy a varázsló hozza létre a menüt és az eszköztárat. Ezzel egy keretet (a szokásos függvényekkel és csatolásokkal) kapunk, amelyet remekül tudunk majd használni.
A ***Next>*** gomb megnyomása után az eszköztár létrehozásával kapcsolatos lapra kerülünk. Itt a ***File*** kategóriából választjuk ki az ***Open*** és a ***Save*** pontokat, majd adjuk hozzá az eszköztárhoz a jobbra mutató nyíl segítségével. Ezután ugyanígy adjuk hozzá az ***Edit*** kategóriából az ***Undo***, ***Redo***, ***Cut***, ***Copy*** és ***Paste*** pontokat, valamint rendezzük el az eszköztárat elválasztókkal (***Separators***), ahogy a 3. *képen* is látható. Az utolsó oldalon semmit nem kell tennünk, csupán bezárnunk azt. Ezután a Designer a 4. *képen* hasonló állapotot mutatja.
Ahhoz, hogy kívánt programunk elkészüljön, összesen két további dologra van szükségünk: egyrészt a főablak neve ne Form1, hanem valamilyen értelmes név legyen, másrészt pedig szükségünk van a szerkesztőmezőre, hisz anélkül bajosan tudunk szöveget szerkeszteni. Az első igény könnyen kielégíthető: a ***Property Editor*** ablakban a ***Properties*** lapon kell körbenéznünk. Ez a lap mindig az éppen használt elem tulajdonságait és azok értékeit kínálja fel szerkesztésre. Mivel utoljára a Form1 formmal dolgoztunk, nem kell külön rákattintanunk. A ***name*** tulajdonság értékét változtassuk meg `ljeditor`-ra, ezáltal nevet adunk az osztálynak, majd a ***caption*** mezőbe írjuk be az ablak fejlécében kívánt szöveget (a változatosság kedvéért legyen `ljeditor`).
Most hozzuk létre a szövegszerkesztő területet. Kattintsunk a ***Richtext Editor*** ikonra (a 4. képen a jobb szélétől számolva az ötödik ikon, „abc de” felirattal), majd kattintsunk az



6. kép Az italics bekapcsolására az ljedit dőlt betűket használ majd

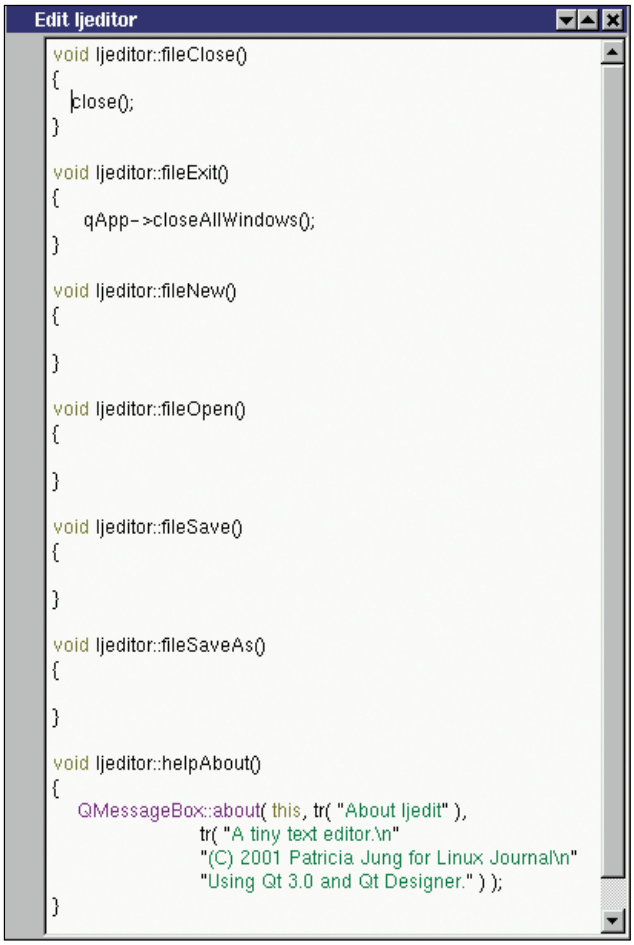


7. kép Új függvényváz létrehozása

ljeditor form pöttyözött hátterén. Az új szerkesztőmezőt a szokásos módon méretezhetjük. Kereszteljük el `TextEdit` névre. Már csak a program szolgáltatásait kell biztosítanunk. Először is távolítsuk el a nem használandó tevékenységeket (***Actions***) az ***Action Editor*** segítségével (*5. ábra*). Töröljük az ollót ábrázoló ikonnal (vagy a helyi menüből kiválasztott ***Delete*** ponttal) az alábbi elemeket: ***helpIndexAciton***, ***helpContentsAction***, ***editFindAction***.

Tevékenységek

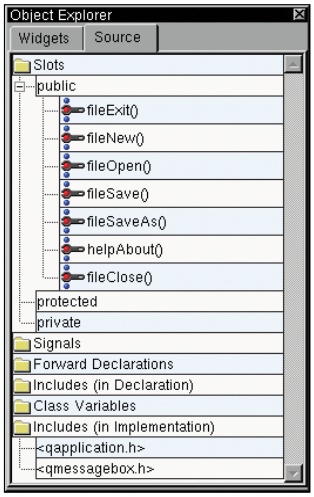
A szerkesztőnek szüksége van néhány további tevékenységre is, hogy kezelni tudjuk a szöveg formázását (félkövér, dőlt, aláhúzott). Az ***Action Editor*** ablakban lévő ***New*** lenyíló menüből (Egy üres lap) válasszuk a ***New Dropdown Action Group*** pontot, melybe majd a módosítók kerülnek, és állítsuk be fontosabb értékeit: a name tulajdonsághoz írjuk be a ***fontCharacter*** nevet, majd az ***iconSet*** tulajdonságra kattintva a megjelenő „...” gomb segítségével válasszuk ki a megfelelő ikont. Az ***Add...*** gombbal külső fájlt is adhatunk hozzá.
A ***menuText***-hez és a súgószövegekhez írjuk be a „Font Characteristics” szöveget. A ***tooltip***-hez és a ***statustip***-hez írjuk be: „Choose font characteristics”, valamint fontos, hogy az ***exclusive*** tulajdonság értékét ***False***-ra állítsuk. Ezzel érjük el,



8. kép Egyszerű forráskódsorokkal ilyen könnyű a szolgáltatásvázat kiegészíteni

hogy a felhasználó egyszerre több tulajdonságot is ki tud választani (például egy szövegrészt félkövérre és dőltre formázhat). Az ***exclusive*** értékét ***True***-n hagyva egyszerre csak az egyik elemet lehet kiválasztani.
A ***fontCharacter*** csoporton jobb gombot nyomva, majd a ***New Action*** (új tevékenység) menüpontot választva a helyi menüből, adjunk hozzá egy ***italics*** nevű tevékenységet a CTRL+I gyorsbillentyűvel, ***Italics*** szöveggel. Mivel a felhasználó ezt a tulajdonságot be és ki tudja kapcsolni, fontos, hogy a ***toggleAction*** értékét ***True***-ra állítsuk. Alapértelmezésként nem szeretnénk dőlt szöveget, ezért az ***on*** tulajdonságot kapcsoljuk ki.
A másik két gyermektevékenységet is hozzuk létre, mind a félkövér (bold, CTRL+B), mind pedig az aláhúzott (***underline***, CTRL+U) számára.
Az elkészített ***fontCharacter*** csoportot egyszerűen úgy adjuk hozzá az eszköztárhoz, hogy megfogjuk az ***Action Editor***-ban, és áthúzzuk az eszköztárra. Ha elé vagy mögé elválasztót szeretnénk, a kívánt helyen kattintsunk a jobb gombbal, majd a helyi menüből válasszuk az ***Insert Separator*** pontot.
Ha most átlépünk előnézetbe (CTRL+T), láthatjuk, hogy az elkészített elemek egyelőre semmit sem csinálnak. Hogy életre keltsük őket, ismét visszatérünk az ***Action Editor*** ablakba, kiválasztjuk az italics elemet, majd a piros-kék ***Edit Connections*** (Kötések szerkesztése) gombra kattintva kiválasztjuk a `toggled(bool)` jelet a ***Signals*** listából. Ahelyett hogy az `ljeditor` egyik foglalatához (slot) kötnénk, a ***Slots*** lenyíló

listából a `TextEdit`-et választjuk, majd az alatta lévő listából kiválasztjuk a `setItalics(bool)` elemet (ez egyébként a `QTextEdit` osztályhoz tartozik, a `TextEdit` is ennek az osztálynak a tagja). Itt semmi más dolgunk nincs, a kötés létrejött (ahogy az ablak alsó részében megjelenő új sor ezt jelzi is), zárjuk be az ablakot az ***Ok*** gombbal. Ugyanezzel az eljárással a ***bold*** tevékenységet kössük a `setBold(bool)` foglalathoz, valamint az ***underline*** tevékenységet a `setUnderline(bool)` foglalathoz. Ha ezzel megvagyunk, próbáljuk ki ismét az előnézetet. Igen! Programunk kezeli a CTRL+I, CTRL+B, CTRL+U gyorsbillentyűket!



9. kép Az Object Explorerrel könnyű új fejállományokat hozzáadni

Most pedig jöjjenek a beépített tevékenységek kapcsolatai! Ezeket nem lehet „váltani”, tehát a `toggled(bool)` jel nem jön szóba. Ehelyett el tudnak indítani egyszerű parancsokat, mint például „Mentsd az adatokat”. Ezért az `activated()` jelet kell a megfelelő foglalatba kötnünk.

Az `edit RedoAction` számára ez a `TextEdit::redo()`. Leválasztjuk tehát ezt a tevékenységet az `ljeditor::editRedo()`-ról. Ezt egyébiránt akkor használtuk volna, ha nem akarnánk a `QTextEdit Redo()`-jára támaszkodni. Ugyanígy, az `editUndoAction action()` jelét kössük a `TextEdit::Undo()`-hoz, és válasszuk le az `ljeditor`-ban lévő vázfüggvényről. Ugyanígy járjunk el az `editPasteAction` és a `TextEdit::Paste()` foglalat, az `editCopyAciton` és a `TextEdit:copy()` foglalat, valamint az `edit CutAction` és a `TextEdit::Cut` foglalat esetén.

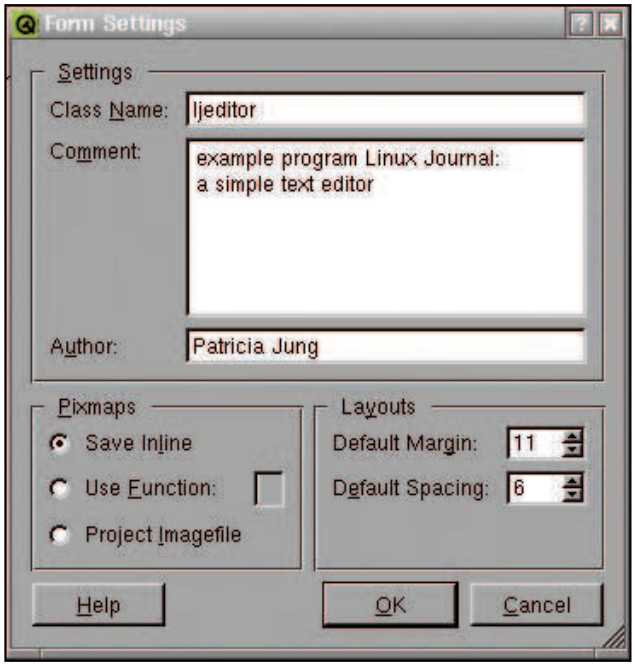
A további előre elkészített tevékenységek (`helpAboutAction`, `fileExitAction`, `fileSaveAction`, `fileSaveAsAction`, `fileOpenAction` és a `fileNewAction`) továbbra is az `ljeditor`-ban megadott vázhoz kapcsolódjanak, ezeket a későbbiek során még programkóddal egészítjük ki.

Egy fontos művelet azonban továbbra is hiányzik, nevezetesen az, amelyet a felhasználó akkor indít el, amikor az éppen használt szerkesztőablakot szeretné bezárni a `fileExitAction` művelethez hasonlóan (ez utóbbi a teljes alkalmazásból kilép).

A munkafolyamat immár ismerős: a tevékenységszerkesztőben kiválasztjuk a ***Create New Action*** menüpontot, majd a ***Property Editor***-ban a `fileCloseAction` nevet adjuk neki, begépeljük a „Close” szöveget; gyorsbillentyűnek pedig a CTRL-Z-t választjuk.

Mindazonáltal az `accelerated()` függvény kötési helye még mindig hiányzik. E hiányosság kijavításához a ***Designer Edit*** menüjében válasszuk a ***Slots*** pontot. A *7. képen* látható a megjelenő ablak.

Hozzunk létre egy új foglalatot `fileClose()` függvénynévvel, visszatérési típusa (***return type***) legyen üres (***void***), elérése (***Acess*** type) pedig public. Az ***Edit Slots*** párbeszédablak nem képes csodákat művelni, csupán egy függvényvázat hoz létre az `ljeditor` osztálymeghatározásán belül. Ezek után hozzákötheti jük a `fileCloseAction`



10. kép Hadd tudja meg a világ, ki készítette ezt a programot!

`activated()` jelét az `ljeditor::flieClose()`-ához az ***Action Editor*** már ismert ***Edit Connections*** ablakában. Mivel ezt a szolgáltatást csak menün keresztül szeretnénk használni, egyszerűen fogjuk meg és húzzuk rá az `ljeditor` formon található ***File*** menüre.

További pár sor kód

A Qt alatt egy elem bezárása egyszerű: minden Qt-elem örökli a `close()` függvényt a Qt-elemek „ősatyjától”, a `QWidget`-től. Ez nem jelent komoly programozási feladatot, így elegendő mindössze egyetlen sort írni a `fileClose()` függvénybe. Miután a `ljeditor` formon jobb gombbal kattintottunk, megjelenik a helyi menü. Ennek Source pontja lesz a segítségünkre. A menüpont választáskor megjelenik a forráskód-szerkesztő ablak, itt már könnyedén beszúrhatjuk egysoros programunkat (*8. kép*).

```
void ljeditor::fileClose( )
{
      close( );
}
```

A `fileExit()` függvényt is egyből megírhatjuk. Az alkalmazásból való kilépéshez az alkalmazásobjektum `closeAllWindows()` függvényét kell meghívnunk:

```
void ljeditor::fileExit( )
{
      qApp->closeAllWindows( );
}
```

Minthogy a Designer magukkal a `QApplication` objektumokkal rendszerint nem foglalkozik, a `qapplication.h`-ra (mely tartalmazza a `qApp` proxyt az valódi alkalmazásobjektum eléréséhez) alapértelmezésben nincs hivatkozás, így az ***ui*** elkészítésekor létrejövő kód nem fordul le.

Tipptár

Tevékenységek
A mai grafikus programok használata közben gyakran megtörténik, hogy valamilyen tevékenységet el akarunk végeztetni, például a fájl mentését, egy ablak bezárását stb. Ezeket a tevékenységeket általában többféleképpen elérhetjük: menüből, gyorsbillentyűvel, az eszköztár egy gombjával. Hogy ne kelljen többször megírnunk ugyanazt a szolgáltatást, az egy-egy szolgáltatáshoz tartozó szükséges elemeket összegyűjtünk egy úgynevezett tevékenységben (action). Ahelyett, hogy a menü egyik pontjaként vagy egy eszköztárelemként írnánk meg ugyanazt, az elkészített tevékenységet egyszerűen felrakjuk az eszköztárra vagy a menübe, az pedig a környezet igényeinek megfelelően menüpontként, gombként vagy egyéb elemként viselkedik.

Jelek és foglalatok
Az összetevőkre bontható programozás érdekében Qt alatt az alábbi gondolatvilágot követjük: amikor egy objektumnak egy tulajdonsága megváltozik, egy jelet (signal) hoz létre (egy gomb például egy `clicked()` jelet hoz létre, amikor az állapota „üres”-ről a „kattintottak rajtam”-ra változik). Más objektum az erre felkészített saját függvényeit, úgynevezett foglalatait (slots) ráirányíthatja az őt érdeklő jelekre. Így például az eszköztár Megnyitás gombja által készített jelet hozzáköthetjük a `fileOpen()` foglalathoz. Mivel a jelek és a foglalatok nem részei a szabványos C++ nyelvnek, megfelelő formára alakításukhoz egy `moc` (meta object compiler) nevű előfeldolgozó szükséges. Ezért bonyolultak még az egyszerű Qt-alkalmazásokhoz készített `Makefile`-ok is.

Súgószövegek
Az állapotszöveg (statustip) és az eszköztipp (tooltip) két olyan szöveg, amelyekkel a felhasználót segíthetjük. Amikor a felhasználó elidőz egy elem fölött és az adott elemhez súgószövegeket adtunk meg, azok a megadott módon megjelennek (az állapotszöveg az állapotsorban, az eszköztipp az egér mellett, általában egy kis sárga ablakban).

Szerencsére egy ***Object Explorer*** (objektumtallózó) a segítségünkre siet a fejállományokra hivatkozás elkészítésekor is. Alapban a ***Widgets*** lap látszik rajta (lásd a 4. képen), de nekünk a ***Source*** lapra van most szükségünk. Kattintsunk ezen a lapon jobb gombbal az ***Includes (in Implementation)*** ponton, majd válasszuk a ***New*** pontot. Ne feledjük, hogy a fejállomány nevét <> jelek közé kell tennünk: `<qapplication.h>` (lásd a *9. képet*). Egy másik foglalat, amit anélkül tölthetünk meg tartalommal, hogy fordítási hibák tömegétől kelljen félnünk, a `helpAbout()`. Ez akkor kerül meghívásra, ha a felhasználó megnyitja a Névjegy (***About***) ablakot, és egy egyszerű párbeszédablakot jelenít meg „About ljedit” címmel és némi szöveggel. Ha a `tr()` függvényt használjuk a szövegek kiíratásakor, a honosítási kérdéseknél lényegesen kevesebb gonddal kell majd szembenéznünk (ezzel a témakörrel a következő számban foglalkozunk):

```
void ljeditor::helpAbout( )
{
    QMessageBox::about( this, tr( "About
        ➥ljedit" ),
        tr( "A tiny text editor.\n"
        "(C) 2002 Patricia Jung for Linux
        ➥Journal\n"
        "Using Qt 3.0.4 and Qt Designer." )
);
}
```

A `QMessageBox::about()` használatához a forráskódba be kell illesztenünk a `<qmessagebox.h>` fejállományt, ahogyan ezt a `<qapplications.h>` állománnyal is tettük. A maradék működésbeli kiegészítést alosztály formájában a következő alkalommal fogjuk elkészíteni.
Ezek után a Designerrel nincs más teendőnk, mint „letisztítani” az új felhasználói felületet.

Fontos csinosítások

Mielőtt bezárjuk a Designert, ellenőrizzük, hogy valamelyik gyorsbillentyűt nem használtuk-e fel kétszer. Ezt az ***Edit*** menü ***Check Accelerators*** pontjának kiválasztásával tehetjük meg. Továbbá eltávolítjuk az összes `ljeditor` foglalatot, melyet nem fogunk a későbbiekben sem használni, vagy kiváltottuk egy `TextEdit` foglalattal. Az ***Edit*** menü ***Slots*** pontját választva a már ismert ***Edit Slots*** ablakba jutunk. Jelöljük ki az `editUndo()`-t, majd a ***Delete Slots*** gombra kattintva töröljük (*7. kép*), majd teljesítsük be hasonlóképpen az alábbi foglalatok sorsát: `editRedo()`, `editCut()`, `editCopy()`, `editPaste()`, `editFind()`, `helpIndex()`, `helpContents()` és `filePrint()`.
Használjuk a GUI előnézetet, és távolítsuk el a felesleges elválasztókat is. Mivel az `editFindAction`-t nem használjuk, egy árva elválasztó található az `ljeditor` ***Edit*** menüjének alján. Valamikor régen volt még egy ***Find*** pont ez alatt. Az elválasztó törléséhez a jobb gombbal kattintsunk rá a formon, majd a helyi menüből válasszuk a ***Delete*** pontot. Ugyanígy járjunk el a ***File*** menüben is a felesleges elválasztókkal.
Minden elem a helyén van, most már a Qt dolga, hogy elhelyezze őket. Jelöljük ki az egész formot, majd válasszuk a ***Layout*** menü ***Lay Out Vertially*** pontját. Ha a felhasználó megváltoztatja az ablak méretét, a `TextEdit` mező is követni fogja az ablak változását. Ha további elemeket akarunk rakni a formra, előtte fel kell bontanunk az elrendezést.
És még egy dolog. Ezt a programot a sajátunkként szeretnénk terjeszteni. Töltsük tehát ki az ***Edit*** menü ***Form Settings*** hatására megjelenő ablak ***Author name*** és ***Description*** mezőit. Ezekben azt is megadhatjuk, hogy a programhoz tartozó képeket külön fájlban kívánjuk tárolni, vagy pedig befordíttatjuk őket közvetlenül a felhasználói felület forrásába.
Végül válasszuk a ***File*** menü ***Save All*** pontját, és már készen is vagyunk! Az utolsó lépéssel projektünköz hozzáadtuk a felületet leíró XML-fájlt (nevezzük, mondjuk ***ljeditor.ui***-nak), valamint az ***ljeditor.ui.hu***-t, mely az általunk begépelt kódokat tartalmazza.

Linux Journal 2002. szeptember, 101. szám

Patricia Jung
(trish@trish.de) háttérismereteit rendszergazdai, műszaki szakírói és újságszerkesztői tevékenységéből meríti, és ilyen minőségében jól érzi magát, hogy kiváltságosként a Linuxszal, illetve a Unixszal foglalkozhat.

Kizárási módszerek a rendszermagban

Robert kifejti nekünk a rendszermag-kizárási módszerek működését, hogy miért van rájuk szükség, és hogy hogyan hoznak létre a segítségükkel a rendszermagfejlesztők biztonságos kódot.

A megfelelő kizárás alkalmazása nehéz, sőt, nagyon nehéz. Ha a rendszermagban nem a megfelelő kizárást alkalmazzuk, véletlenszerű lefagyásokban lesz részünk, illetve egyéb furcsaságok fordulhatnak elő. Rosszul megválasztott kizárások alkalmazásakor a kód nehezen olvashatóvá válik, a működése sem lesz megbízható, és a rendszermagon dolgozó társaidat is az őrületbe kergeted. Ebben a cikkben kifejtem, miért szükséges a rendszermagban kizárásokat alkalmazni. Felsorakoztatok néhány általános kizárási szabályt, végül pedig kitérek az alkalmazható kizárási módozatokra.

Miért szükséges kizárást alkalmazni a rendszermagban?

Kizárások alkalmazása elsősorban a rendszermagban zajló folyamatok összehangolásához szükséges. Ezekhez a folyamatokhoz – vagyis a kényes szakaszokhoz – szükség van bizonyos fokú védelemre, amellyel az események megfelelő időben történő kezelése, illetve az események többszöri kezelésének elkerülése biztosítható. Megfelelő kizárások alkalmazása nélkül úgynevezett versenyhelyzet jön létre. Képzeld csak el, hogy egy közösen használt osztott `i` változó esetén egy `i++` is milyen veszélyes lehet! Tegyük fel, hogy az értékét először az egyik processzor olvassa ki, majd a másik, utána pedig mindketten növelik az értékét, és mindketten visszaírják a memóriába. Ha az `i` értéke eredetileg kettő volt, akkor a műveletet követően négynek kellene lennie, csakhogy így ez az érték valójában mindössze ***három*** lesz!

Ez nem azt jelenti, hogy kizárási gondok csak SMP-rendszereken jelentkezhetnek (SMP – egyidejűleg több processzorral dolgozó rendszer). Megszakításkezelők is vethetnek fel kizárási gondokat, akárcsak a megszakításos ütemezőn alapuló rendszermagok, vagy bármely kód, ami alvó módba tér. Ezek közül csak az SMP tekinthető teljesen egyidejűnek, mivel egyedül SMP-rendszereken fordulhat elő, hogy két vagy több kódrészlet teljesen azonos időben fut. Az első esetekben is felléphetnek látszólagos egyidejűség, ahol – bár a kódrészletek nem teljesen azonos időben futnak le – mégis összezavarhatják egymás adatait.

E kényes kódrészletek esetén mindenképpen szükséges a kizárások alkalmazása. A Linux-rendszermag több kizáró módszert is kínál, így biztosítva a fejlesztőknek a biztonságos és hatékony kód megírásához szükséges feltételeket.

SMP-kizárások egyprocesszoros magokban

Függetlenül attól, hogy többprocesszoros rendszert használsz-e vagy sem, elképzelhető, hogy azok az emberek, akik a kódodat futtatják, ilyen rendszert használnak. Ennek következtében az olyan kód, amely a kizárásokat nem megfelelően kezeli, nem kerülhet be a Linux rendszermagjába. Megszakításosan ütemező rendszer esetén egyprocesszoros rendszereken is létkérdés a megfelelő kizárások alkalmazása. Ezért ne feledd: a kizárások alkalmazása rendkívül fontos.

Linus nagyszerű döntésének köszönhetően az SMP- és egyprocesszoros rendszermagok szerkezete eltér egymástól. Ennek következtében bizonyos kizárások egyáltalán nem léteznek az egyprocesszoros rendszermagokban. A `CONFIG_SMP` és `CONFIG_PREEMPT` egészen más kizárási módozatokat eredményezhet. Bár mindez a fejlesztő számára teljesen mindegy: mindíg alkalmzzunk kizárást, így semmilyen helyzetben nem lehet gond.

Oszthatatlan műveletek

Először az oszthatatlan műveleteket vesszük át, két okból is. Elsősorban azért, mert ezek jelentik a legegyszerűbb időzítést a rendszermagban, így ezek használata érhető meg a legegyszerűbben. Másodsorban pedig azért, mert az összetett kizárási módszerek is az egyszerű időzítéseken alapulnak. Ilyenformán ezek rendszermagkizárásokat létrehozó blokkokat alkotnak.

Az oszthatatlan műveleti jelek, mint az összeadás vagy a kivonás, egyetlen oszthatatlan műveletet alkotnak. Vegyük az előző `i++`-os példát. Ha képesek lennénk kiolvasni az `i` változót, az értékét növelni, majd egyetlen oszthatatlan műveletben visszaírni a memóriába, nem alakulhatna ki az előzőleg említett versenyhelyzet. Az oszthatatlan műveleti jelek segítségével atomi, vagyis oszthatatlan műveleteket végezhetünk el.

Két fajtájuk létezik: az egész számokkal dolgozó tagfüggvények és a bitekkel dolgozó tagfüggvények. Egész számok esetén a szükséges kód így néz ki:

```
atomic_t v;

atomic_set(&v, 5);  /* v = 5 (oszthatatlan) */
atomic_add(3, &v);  /* v = v + 3
                     ➥(oszthatatlan) */
atomic_dec(&v);     /* v = v - 1
                     ➥(oszthatatlan) */
printf("Az eredmØny 7 lesz: %d\n",
      ➥atomic_read(&v));
```

Ez elég egyszerű. Létezik viszont néhány kikötés, amelyeket az oszthatatlan műveletek alkalmazásakor figyelembe kell vennünk. Először is az `atomic_t` típussal létrehozott változókat csak és kizárólag az `atomic`-függvények képesek kezelni. És ugyanez fordítva igaz, vagyis az `atomic`-függvények csakis `atomic_t` típusú változókat képesek kezelni. Végül pedig az egyes rendszerek közötti különbségek miatt az `atomic_t` típusú változónak csak az első 24 bitjét vehetjük biztosnak. A „Függvényreferencia” részben az oszthatatlan műveletekkel kapcsolatos összes függvényt megtalálod.

Az oszthatatlan műveletek következő fajtája az, amely különálló bitekkel dolgozik. Az ilyenek egyszerűbbek, mint az egész számra épülő tagfüggvények, mivel képesek a C beépített

típusaival dolgozni. Vegyük például a `void set_bit(int nr, void addr)` függvényt. Ez a függvény az `addr` által mutatott adat `nr`-edik bitjét 1-esre állítja. Ezek a függvények is megtalálhatók a „Függvényreferencia” részben.

Forgózárak

Bármilyen más esetben, ami egy picit is bonyolultabb, mint a fenti példa, sokkal teljesebb megoldásra lesz szükségünk. Az egyik legegyszerűbb kizárási módozat a forgózár (spinlock), ami az ***include/asm/spinlock.h*** és ***include/linux/spinlock.h*** fájlokban van meghatározva. A forgózár egy nagyon egyszerű fenntartó kizárás. Ha egy folyamat egy forgózárat próbál megszerezni, de az foglalt, a folyamat addig kénytelen várni, míg a forgózár fel nem szabadul. Így egyszerű és gyors kizárás jön létre. Lássunk egy példát a forgózárak legegyszerűbb fajtájára!

```
spinlock_t mr_lock = SPIN_LOCK_UNLOCKED;
unsigned long flags;

spin_lock_irqsave(&mr_lock, flags);
/* kØnyes szakasz... */
spin_unlock_irqrestore(&mr_lock, flags);
```

A `spin_lock_irqsave()` használatával helyileg letilthatjuk a megszakítások kezelését, és SMP-rendszereken forgózárakat hozhatunk létre. Ilyen módon mind a megszakításokból, mind az SMP-ből adódó egyidejűségeket kivédhetjük. A `spin_unlock_irqrestore()` függvény meghívásával a megszakítások az eredeti állapotba állíthatók vissza. Egyprocesszoros rendszermagok esetén a fenti kód a következőképpen fordul le:

```
unsigned long flags;

save_flags(flags);
cli();
/* kritikus szakasz... */
restore_flags(flags);
```

Ez a kódrészlet anélkül biztosítja a szükséges megszakítások elleni védelmet, hogy feleslegesen SMP-védelmet is biztosítana. Ennek a védelemnek egy másik fajtája a `spin_lock_irq()`. Ez a függvényhívás feltétel nélkül letiltja a megszakításokat, hasonlóan a `cli()` és `sti()` függvényekhez. Például:

```
spinlock_t mr_lock = SPIN_LOCK_UNLOCKED;

spin_lock_irq(&mr_lock);
/* kØnyes szakasz... */
spin_unlock_irq(&mr_lock);
```

Ez a kódrészlet csak akkor biztonságos, ha a megszakítások eredetileg – még a kizárás megszerzése előtt – sem voltak letiltva. Mivel a rendszermag egyre nagyobbá és bonyolultabbá válik, a rendszermagbeli szálak is egyre követhetetlenebbek lesznek, így az effajta kizárásokat ajánlatos messze elkerülni. Kivéve természetesen, ha nagyon biztos vagy a dolgodban.
A fentebb használt forgózárak feltételezik, hogy az így védett adat mind a megszakításkezelőkben, mind pedig magmódban elérhető. Ha biztos vagy benne, hogy az adott kód csak a felhasználó szintjén hívódik meg (például rendszerhívások esetén), akkor elegendő a `spin_lock()` és `spin_unlock()` függvények használata, amelyek egy adott kizárást foglalnak le vagy szabadítanak fel, a megszakításokra azonban nincsenek hatással. A forgózárak legutolsó változata a `spin_lock_bh()` függvény, mely az alapvető kizárást biztosítja és a `softirq`-kat tiltja le. Erre olyan kódrészleteknél van szükség, melyek többnyire `softirq`-kon kívül futnak, de alkalmanként `softirq`-ból is futhatnak. Az ehhez tartozó függvény a `spin_lock_bh()`.
Fontos, hogy Linuxban a forgózárak nem ismételten meghívhatók, mint néhány más operációs rendszer esetén. Sokak szerint erre egyébként sincs szükség, mivel a többszörös forgózármeghívás csak a kód minőségét rontaná. Ez annyit tesz, hogy ügyelned kell, nehogy olyan forgózárat foglalj le, amelyet már tartasz, mivel így a program könnyen holtpontra juthat.
A forgózárak alkalmazása olyan helyeken szükséges, ahol a kizárás használata csak rövid ideig kívánatos, mivel a folyamat addig vár tétlenül, amíg a kizárás fel nem szabadul (a „Szabályok” szakaszra pillantva megtudhatod, milyen várakozások minősülnek túl hosszúnak). Szerencsére a forgózárak bárhol használhatók, kivéve az olyan kódrészletet, ami bizonyos esetekben alvó módba kerülhet. Soha ne hívj meg olyan kódot, amely a felhasználó memóriáját kezeli, kerüld el a `kmalloc()` függvényt a `GFP_KERNEL` zászlóval (flag), valamint a jelzőkkel (semaphore) és az ütemezéssel kapcsolatos függvényeket is felejtsd el, ha bármilyen foglalt forgózárral rendelkezel.
Ne feledd, én figyelmeztettelek!
Ha olyan kizárást szeretnél, amelyet akár hosszabb ideig is tarthatsz, és nem zavarja a szálak egyidejűsége, de az esetleges alvó mód sem, akkor a jelzőkre van szükséged.

Jelzők

A jelzők (semaphore) a Linuxban alvó kizárások. Ha a sor egy másik folyamattal versenyhelyzetbe kerül, a jelzők a folyamatot alvó módba küldik. A forgózárakkal ellentétben jelzőket akkor használunk, ha a rájuk való várakozás hosszabb ideig is elhúzódhat. Rövidebb várakozások esetén viszont a jelzők használatával a processzoridőt pazaroljuk, mivel ilyenkor számolnunk kell azzal, hogy egy jelzőre várakozó folyamat alvó módba tér, amit azután, ha a jelző felszabadult, fel kell ébreszteni – és ez elég költséges folyamat. Mindazonáltal mivel a szál ilyenkor alvó módba tér, a jelzők olyan helyeken is használhatók, ahol a forgózárak nem. Más szóval jelzők tartásakor az adott kódrészlet futását akár blokkolhatjuk is.
Linuxban a jelzőket egy C-szerkezet testesíti meg: a ***include/asm/semaphore.h*** fájlban található `struct semaphore`. Ez a szerkezet egy olyan mutatót foglal magában, amely egy várakozási sorra mutat, illetve egy felhasználásszámlálót is tartalmaz. A várakozási sor azoknak a folyamatoknak a listáját tartalmazza, amelyek a jelzőre várakoznak. A felhasználásszámláló az egyidejűleg tartható folyamatok számát mutatja meg. Ha ez az érték negatív, a jelző nem érhető el, és a számláló abszolút értéke a várakozási sorban tartózkodó folyamatok számát tartalmazza. Ennek a számlálónak futás közben a `sema_init()` függvény ad kezdeti értéket. Ez többnyire `1` (ilyen esetben a jelzőt `mutex`-nek, vagyis kölcsönös kizárásnak hívjuk).
A jelzőhöz két függvényen keresztül férhetünk hozzá: a `down`-on és az `up`-on keresztül (vagyis fel és le, ezeket történeti okokból `P`-nek és `V`-nek is nevezzük). Az első megpróbálja lefoglalni a jelzőt, és ha ez nem sikerül, várakozik; a másik ellenben felszabadítja a jelzőt és vele együtt minden várakozó folyamatot.
A Linuxban a jelzők használata egyszerű. A jelző lefoglalásához

Függvényreferencia

Forgózárak

- `void spin_lock(spinlock_t *lock)`: a megadott kizárást lefoglalja, és ha foglalt, addig várakozik, amíg fel nem szabadul.
- `void spin_lock_irq(spinlock_t *lock)`: hasonló a `spin_lock()`-hoz, de a helyi processzoron a megszakításokat is letiltja.
- `void spin_lock_irqsave(spinlock_t *lock, unsigned long flags)`: ugyanaz, mint a `spin_lock_irq()`, viszont a megszakításjelzőket menti a `flags` változóba.
- `void spin_lock_bh(spinlock_t *lock)`: olyan, mint a `spin_lock()`, de a `softirq`-k futtatását is megakadályozza.
- `void spin_unlock(spinlock_t *lock)`, `void spin_unlock_irq(spinlock_t *lock)`, `void spin_unlock_irqrestore(spinlock_t *lock, unsigned long flags)` és `void spin_unlock_bh(spinlock_t *lock)`: az előző függvények felszabadító változatai.
- `void spin_trylock(spinlock_t *lock)`: ha a kizárás foglalt, nullától különböző értékkel tér vissza, máskülönben a visszaadott érték nulla.
- `void read_lock(rwlock_t *lock)` és `void read_unlock(rwlock_t *lock)`: az olvasó–író kizárás olvasó változatát foglalja le és szabadítja fel.
- `void write_lock(rwlock_t *lock)` és `void write_unlock(rwlock_t *lock)`: az olvasó–író kizárás író változatát foglalja le és szabadítja fel.

Jelzők

- `void sema_init(struct semaphore *sem, int val)`: kezdeti értéket ad a jelző felhasználásszámlálójának.
- `void down(struct semaphore *sem)`: az adott jelzőt próbálja megszerezni, vagy a felszabadulására vár.
- `int down_interruptible(struct semaphore *sem)`: mint a `down()`, de jel esetén `EINTR`-rel tér vissza.
- `int down_trylock(struct semaphore *sem)`: mint a `down()`, de ha a jelző foglalt, azonnal visszatér egy nullától különböző értékkel.
- `void up(struct semaphore *sem)`: felszabadítja az adott jelzőt.
- `void init_rwsem(struct rw_semaphore *rwsem)`: az író–olvasó jelzőnek kezdeti értékül `1`-et ad.
- `void down_read(struct rw_semaphore *rwsem)` és `void up_read(struct rw_semaphore *rwsem)`: olvasás esetén az adott olvasó–író jelzőt foglalja le, illetve szabadítja fel.
- `void down_write(struct rw_semaphore *rwsem)` és `void up_write(struct rw_semaphore *rwsem)`: az adott olvasó–író jelzőt írás esetén foglalja le, illetve szabadítja fel.

Nagy rendszermagkizárás

- `void lock_kernel()`: teljes kizárás igénylése (BKL).
- `void unlock_kernel()`: a BKL felszabadítása.
- `int kernel_locked()`: ha a BKL foglalt, igaz értékkel tér vissza, máskülönben hamissal.

Megszakításos ütemezés tiltása

- `void preempt_disable()`: növeli a megszakításos ütemezés számlálóját.
- `void preempt_enable()`: csökkenti a megszakításos ütemezés számlálóját, és engedélyezi a megszakításos ütemezést, ha szükséges.
- `void preempt_enable_no_resched()`: csökkenti a megszakításos ütemezés számlálóját.
- `int preempt_get_count()`: lekérdezi a megszakításos ütemezés számlálóját.

a `down_interruptible()` függvényt kell meghívni, amely eggyel csökkenti a jelző felhasználásszámlálóját. Ha az új érték negatív, a folyamat a várakozási sorba kerül. Amennyiben az új érték nullával egyenlő vagy nagyobb, a folyamat megkapja a jelzőt, és a függvény `0` értékkel tér vissza. Ha a várakozást valamilyen jel szakítja meg, a hívás `-EINTR`-rel tér vissza a jelző megszerzése nélkül.
Az `up()` függvénnyel a jelzőt a felhasználásszámláló értékét növelve felszabadíthatjuk. Amennyiben az új érték `0` vagy nagyobb, a várakozási sorban lévő folyamatok némelyike felszabadul:

```
struct semaphore mr_sem;

sema_init(&mr_sem, 1);      /* a felhasznÆlÆs-
                            ↪szÆmlÆl  1 */

if (down_interruptible(&mr_sem))
  /* a jelzı lefoglalÆsa egy jelzØs miatt */
    / * nem val sult meg... */

/* kØnyes szakasz (a jelzıt lefoglaltuk) ... */
up(&mr_sem);
```

A rendszermag egy `down()` függvényt is biztosít, ami annyiban különbözik a `down_interruptible()` fügvénytől, hogy a folyamatot meg nem szakítható alvó módba küldi. Ebben az állapotban minden jelzést, amit a folyamat kap, figyelmen kívül hagy. A programozók többnyire a `down_interruptible()` függvényt használjak. Végezetül pedig a Linux a `down_trylock()` függvényt is felkínálja, ami a jelző foglalt állapotában valamely nullától különböző értékkel blokkolás nélkül tér vissza, ha pedig a jelző szabad, lefoglalja azt.

Olvasó–író kizárások

A forgózárakon és jelzőkön kívül a Linux-rendszermag úgynevezett olvasó–író változatokat is biztosít, amelyek a kizárásokat két csoportba osztják: olvasásra és írásra. Ugyannak az

adatnak az egy időben történő olvasása többnyire nem jár gonddal, legalábbis amíg az adat nem módosul, viszont az írásról nem mondható el ugyanez. Ezért az író–olvasó kizárásokkal engedélyezett a több szálon történő olvasás, írás azonban csak egyetlen szálon lehetséges. Írás közben az olvasás sem engedélyezett. Ha a felhasznált adat egyértelműen kezelhető olvasásokkal és írásokkal, különösen, ha olvasás lényegesen többször fordul elő, akkor az író–olvasó kizárások használata ajánlott.
Az író–olvasó forgózárakat rwlock-nak hívjuk. Használatuk a forgózárakéhoz hasonló, kivéve természetesen az írások és olvasások külön történő kezelését.

```
rwlock_t mr_rwlock = RW_LOCK_UNLOCKED;

read_lock(&mr_rwlock);
/* kØnyes szakasz (csak olvasÆs)... */
read_unlock(&mr_rwlock);

write_lock(&mr_rwlock);
/* kØnyes szakasz (olvasÆs Øs  rÆs)... */
write_unlock(&mr_rwlock);
```

Hasonlóképpen az olvasó–író jelzőket `rw_semaphore`-nak nevezzük. Működésük a rendes jelzőkéhez hasonló, kivéve az írások és olvasások külön történő kezelését:

```
struct rw_semaphore mr_rwsem;

init_rwsem(&mr_rwsem);

down_read(&mr_rwsem);
/* kØnyes szakasz (csak olvasÆs)... */
up_read(&mr_rwsem);

down_write(&mr_rwsem);
/* kØnyes szakasz (olvasÆs Øs  rÆs)... */
up_write(&mr_rwsem);
```

Az olvasó–író kizárások megfelelő alkalmazása érezhető sebességnövekedést eredményez. Megjegyzendő, hogy néhány más rendszerrel ellentétben a Linuxban az olvasási kizárásból nem képezhetünk önműködően írási kizárást. Ugyanakkor ha létezik egy olvasási kizárásunk, és ilyenkor próbálunk kizárólagos hozzáféréshez jutni, az eredmény holtpont lesz. Ha tudod, hogy írásra is szükséged lesz, alapesetben már a művelet legelején írási kizárást kell igényelned. Ha az írások és olvasások közti különbség zavaros, elképzelhető, hogy az író–olvasó kizárások használata nem a legjobb választás.

Erős olvasási kizárások

Az erős olvasási kizárások (vagyis big-reader kizárások, brlockok) az ***include/linux/brlock.h*** fájlban vannak meghatározva. Az effajta kizárások az egyszerű író–olvasó kizárások különleges csoportját képezik. Eredetileg a RedHatnél dolgozó ***Ingo Molnar*** tervezte őket, és olyan forgózárakhoz hasonló szerkezetet hozott velük létre, amivel nagyon gyorsan szerezhetünk engedélyt olvasásra, az írási műveletek esetén ugyanakkor csak rendkívül lassan. Ez a fajta kizárás kifejezetten előnyös az olyan helyzetekben, amikor temérdek olvasószál mellett csak elvétve akad egy-egy írószál. Bár az olvasási kizárások különböznek az egyszerű olvasó–író kizárástól, használatuk megegyezik, eltekintve attól, hogy az erős olvasás kizárások a ***brlock.h*** fájlban található ***brlock_indices*** szakaszban határozódnak meg.

```
br_read_lock(BR_MR_LOCK);
/* kØnyes szakasz (csak olvasÆs)... */
br_read_unlock(BR_MR_LOCK);
```

Az erős olvasáskizárások használata az írások lassúsága miatt csak néhány különleges esetre korlátozott, és ez valószínűleg a jövőben sem fog változni.

Szabályok

Az alábbi szabályok szerint élj, hogy senkinek ne essék baja!

1. Akkor is használj kizárásokat, ha nem használsz SMP-t.
2. A kizárások az adatok kezelésekor és nem a kód futtatásakor szükségesek, vagyis a kizárásoknak az adatokkal kell kapcsolatban állniuk, nem a kódterületekkel.
3. Van egy halvány vonal a durva kizárások és a finom kizárások között is. Egyrészt finom kizárások alkalmazásakor a kizárási verseny csökkentése miatt a kód is hatékonyabb lesz, másrészről minél több kizárást használsz, annál több memória és processzoridő fogy, és a kizárások rendszere is egyre bonyolultabbá válik.
4. Amennyiben a várakozási idő hosszabb, próbáld elkerülni a forgózárak használatát. Ez a hosszabb idő természetesen mindenkinek mást jelent, egy korszerű rendszer esetén az 1–5 milliszekundum tekinthető a felső határnak. Emlékezz vissza, hogy forgózárak esetén az egyes szálak tétlenül várnak. A hosszú idejű kizárások ennek következtében SMP- és megszakításos időosztású rendszereken nagyobb kizárási versenyt alakítanak ki. Jelzők esetén ennek a fordítottja érvényesül: a jelzőre történő várakozáskor a folyamat alvó állapotba kerül, és ezalatt egy másik folyamat kerülhet előtérbe – de ez a művelet lassabb is, így a használata hosszabb várakozási ciklusok esetén indokolt.
5. Kizárási elveidet gondosan tervezd meg, és ragaszkodj is hozzájuk.

A nagy rendszermagkizárás

A 2.0-s rendszermagok óta a Linux egy úgynevezett nagy rendszermagkizárást használ, amelyet korábban az SMP-rendszerekkel való együttműködés kedvéért vezettek be. A 2.2-es és 2.4-es rendszermagok fejlesztésekor rengeteg energiát fektettek e teljes kizárás eltüntetésébe és egy sokkal finomabban szabályozott rendszer kidolgozásába. Ma már a teljes kizárások használata csak elenyésző számban van jelen, de tény, hogy még létezik, így a rendszermag fejlesztőinek tisztában kell lenniük a viselkedésével.
A teljes kizárást nagy rendszermagkizárásnak is hívják, más néven `BKL`-nek. Ez a kizárásfajta többszörösen újrahívható forgózárat takar, amelynek ismételt meghívása sem juttatja holtpontra a rendszert (mint ahogyan az a normál forgózárak esetében történne). Egy folyamat akár alvó állapotba is léphet, sőt még az ütemezőbe is beléphet a `BKL` tartása közben. Ha egy teljes kizárásban lévő folyamat belép az ütemezőbe, a kizárás feloldódik, így más folyamatok szerezhetik meg. A `BKL` e tulaj-

donságainak köszönhető az SMP-rendszerek viszonylag egyszerű kezelése a 2.0-s rendszermagok idejében. Manapság viszont temérdek érv szól már az effajta kizárás alkalmazása ellen.
A nagy rendszermagkizárás használata egyszerű.
A `lock_kernel()` hívással megszerezhető a kizárás, az `unlock_kernel()`-lel pedig elengedhetjük. Hogyha a `kernel_locked()` függvény nullától különböző értékkel tér vissza, a kizárás érvényes, ellenkező esetben nullát kapunk. Lássunk erre is egy példát!

```
lock_kernel();
/* kØnyes szakasz... */
unlock_kernel();
```

A megszakítható ütemezés vezérlése

A 2.5-ös fejlesztői rendszermagoktól kezdődően (illetve egy folttal már a 2.4-es rendszermagok esetében is) a rendszermag ütemezése megszakítható. Ennek a képességnek köszönhetően a rendszermag dönthet bizonyos folyamatok futásának megszakításáról, ezzel magasabb szintű folyamatokat hozhat előtérbe, még akkor is, ha az adott folyamat a rendszermag belsejében fut. A megszakításos ütemezésen alapuló rendszermagok számos, az SMP-rendszerek ütemezéséhez hasonló gondot vetnek fel. Szerencsére a rendszermag az SMP-rendszerekre már fel van készítve, így az SMP-kizárások alkalmazásával a megszakításos ütemezésű rendszermagok is biztonságosan futhatnak. Ennek ellenére érdemes megemlíteni néhány új szabályt. Például kizárással nem védhetünk processzoronként valamilyen processzorhoz kötődő adatot, mivel a védelmük önműködően megy végbe (ez így biztonságos, mivel az ilyen adat minden processzor esetében egyedi), és ez szükséges a rendszermag megfelelő ütemezéséhez.

Összegzés

SMP-rendszerek esetén mind a rendszer megbízhatósága, mind a méretezhetősége folyamatosan nő. Mióta az SMP-kezelés bemutatkozott a 2.0-s rendszermagokban, az egymást követő rendszermagváltozatok e téren rendkívüli mértékben fejlődtek. Ehhez új és okosabb kizárási módszerek bevezetésére, a korábbi kizárási rendszer felülvizsgálatára és a teljes kizárások megszüntetésére volt szükség a gyorsaságot igénylő területeken. Ez a módi a 2.5-ös rendszermagok esetében is folytatódik, így a jövőben bizonyosan még ennél is hatékonyabb rendszermagok születnek.
A fejlesztők ebből úgy vehetik ki a részüket, hogy megfelelő kizárási eljárásokat alkalmaznak, és egyik szemüket mindig a megbízhatóságon és a sebességen tartják.

Linux Journal 2002. augusztus, 100. 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.

IceWM második pillantásra

Marcel Gagné e havi cikkében (78. oldal) röviden kitér egy jól használható és gyors ablakkezelőre, az IceWM-re. Mivel jómagam is nagy rajongója vagyok a gyors és használható megoldásoknak, valamint az IceWM-felhasználók táborába tartozom, egy-két további gondolatot szeretnék hozzáfűzni a leírtakhoz.
Az IceWM tehát akár kisebb gépeken is elfut, mint sok testvére, például a BlackBox. Mindkét ablakkezelő kicsi és gyors. Az IceWM nagy előnye, hogy gyorsan hozzászokhatunk, lényegében a Windowsban is használatos kombinációkat (Alt+Tab: következő ablak, Alt+Esc: háttérbe küld, Ctrl+Esc: IceWM menü stb.) alapból ismeri, illetve néhány rendkívül hasznossal ki is bővíti. Ide tartozik a munkafelületek közötti váltásra használatosak (Ctrl+Alt+BalNyíl, JobbNyíl: előző, illetve következő felület stb.). Külön ügyesnek tartom azt a trükköt, hogy miközben a munkafelületek között váltunk, ha nyomva tartjuk a Shift gombot, az éppen használt ablakot „magunkkal visszük" az új felületre. Ezzel a módszerrel pillanatok alatt rendet teremthetünk.
A Marcel által írt beállításfájlok közül kiemelném a preferences-t, melyben én a következő változásokat eszközöltem: bekapcsoltam az egérgörgő és a Menügombok használatát (UseMouseWheel=1, Win95Keys=1), valamint telepítettem az xexec programot, majd hozzárendeltem az IceWM menüben lévő `Run` parancshoz (`RunCommand="xexec"`), ezt egyébként a baloldali Menü és az R billentyű kombinációjával is előhívható. Ez külön hasznos, ha nem akarunk egy `xterm`-et indítani egy parancs kiadásához (az `xterm` egyébként könnyedén indítható a Ctrl+Alt+T-vel).
Emellett az ablakkezelő egyéb hasznos kényelmi szolgáltatásokkal is rendelkezik, ha egyszer van egy órácskánk, érdemes végigböngészgetni mind a súgót, mind a beállításfájlokat. Érdekes például, hogy a fejlesztők gondoltak a multimédia-billentyűzetek tulajdonosaira is: az XF86 által felismert billentyűkhöz az ablakkezelő alapértelmezett programokat rendel (ezeket a keys fájlban nézhetjük meg).
A BlackBoxról, erről az igazi minimalistáról is érdemes néhány szót szólni. Főleg azoknak ajánlom, akiknek öreg csacsi gépen kell dolgozniuk vagy csillogtatni akarják profizmusukat. Legnagyobb előnye ugyanis, hogy döbbenetesen kicsi és gyors. Önmagában a felület nem túlzottan látványos, egy darab tálcát tartalmaz. Az viszont biztos, hogy nem ütközünk olyan gondokba, hogy például az ablakkezelő elnyeli az Alt+F2 vagy az F11 billentyűket, és hogy e billentyűkombinációkat nem tudjuk a programjainkban használni. Ugyanis a BlackBox semmilyen kombinációt nem ismer. Az egész billentyűkezelő részét különválasztották, és külön bbkeys néven tudjuk futtatni. Ennek segítségével viszont egy egyszerű, mégis nagyszerű felületen keresztül állíthatjuk be (például), hogy az Alt+Tab a szokásos módon működjön. Ugyanitt tetszőleges kombinációhoz bármilyen parancsot könnyedén hozzárendelhetünk. Így egyetlen gombnyomásra indulhat egy `xterm`, az Opera vagy bármi más. Remek felület, csak haladóknak!

Szy György

donságainak köszönhető az SMP-rendszerek viszonylag egyszerű kezelése a 2.0-s rendszermagok idejében. Manapság viszont temérdek érv szól már az effajta kizárás alkalmazása ellen.
A nagy rendszermagkizárás használata egyszerű.
A `lock_kernel()` hívással megszerezhető a kizárás, az `unlock_kernel()`-lel pedig elengedhetjük. Hogyha a `kernel_locked()` függvény nullától különböző értékkel tér vissza, a kizárás érvényes, ellenkező esetben nullát kapunk. Lássunk erre is egy példát!

```
lock_kernel();
/* kØnyes szakasz... */
unlock_kernel();
```

A megszakítható ütemezés vezérlése

A 2.5-ös fejlesztői rendszermagoktól kezdődően (illetve egy folttal már a 2.4-es rendszermagok esetében is) a rendszermag ütemezése megszakítható. Ennek a képességnek köszönhetően a rendszermag dönthet bizonyos folyamatok futásának megszakításáról, ezzel magasabb szintű folyamatokat hozhat előtérbe, még akkor is, ha az adott folyamat a rendszermag belsejében fut. A megszakításos ütemezésen alapuló rendszermagok számos, az SMP-rendszerek ütemezéséhez hasonló gondot vetnek fel. Szerencsére a rendszermag az SMP-rendszerekre már fel van készítve, így az SMP-kizárások alkalmazásával a megszakításos ütemezésű rendszermagok is biztonságosan futhatnak. Ennek ellenére érdemes megemlíteni néhány új szabályt. Például kizárással nem védhetünk processzoronként valamilyen processzorhoz kötődő adatot, mivel a védelmük önműködően megy végbe (ez így biztonságos, mivel az ilyen adat minden processzor esetében egyedi), és ez szükséges a rendszermag megfelelő ütemezéséhez.

Összegzés

SMP-rendszerek esetén mind a rendszer megbízhatósága, mind a méretezhetősége folyamatosan nő. Mióta az SMP-kezelés bemutatkozott a 2.0-s rendszermagokban, az egymást követő rendszermagváltozatok e téren rendkívüli mértékben fejlődtek. Ehhez új és okosabb kizárási módszerek bevezetésére, a korábbi kizárási rendszer felülvizsgálatára és a teljes kizárások megszüntetésére volt szükség a gyorsaságot igénylő területeken. Ez a módi a 2.5-ös rendszermagok esetében is folytatódik, így a jövőben bizonyosan még ennél is hatékonyabb rendszermagok születnek.
A fejlesztők ebből úgy vehetik ki a részüket, hogy megfelelő kizárási eljárásokat alkalmaznak, és egyik szemüket mindig a megbízhatóságon és a sebességen tartják.

Linux Journal 2002. augusztus, 100. 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.

IceWM második pillantásra

Marcel Gagné e havi cikkében (78. oldal) röviden kitér egy jól használható és gyors ablakkezelőre, az IceWM-re. Mivel jómagam is nagy rajongója vagyok a gyors és használható megoldásoknak, valamint az IceWM-felhasználók táborába tartozom, egy-két további gondolatot szeretnék hozzáfűzni a leírtakhoz.
Az IceWM tehát akár kisebb gépeken is elfut, mint sok testvére, például a BlackBox. Mindkét ablakkezelő kicsi és gyors. Az IceWM nagy előnye, hogy gyorsan hozzászokhatunk, lényegében a Windowsban is használatos kombinációkat (Alt+Tab: következő ablak, Alt+Esc: háttérbe küld, Ctrl+Esc: IceWM menü stb.) alapból ismeri, illetve néhány rendkívül hasznossal ki is bővíti. Ide tartozik a munkafelületek közötti váltásra használatosak (Ctrl+Alt+BalNyíl, JobbNyíl: előző, illetve következő felület stb.). Külön ügyesnek tartom azt a trükköt, hogy miközben a munkafelületek között váltunk, ha nyomva tartjuk a Shift gombot, az éppen használt ablakot „magunkkal visszük" az új felületre. Ezzel a módszerrel pillanatok alatt rendet teremthetünk.
A Marcel által írt beállításfájlok közül kiemelném a preferences-t, melyben én a következő változásokat eszközöltem: bekapcsoltam az egérgörgő és a Menügombok használatát (UseMouseWheel=1, Win95Keys=1), valamint telepítettem az xexec programot, majd hozzárendeltem az IceWM menüben lévő `Run` parancshoz (`RunCommand="xexec"`), ezt egyébként a baloldali Menü és az R billentyű kombinációjával is előhívható. Ez külön hasznos, ha nem akarunk egy `xterm`-et indítani egy parancs kiadásához (az `xterm` egyébként könnyedén indítható a Ctrl+Alt+T-vel).
Emellett az ablakkezelő egyéb hasznos kényelmi szolgáltatásokkal is rendelkezik, ha egyszer van egy órácskánk, érdemes végigböngészgetni mind a súgót, mind a beállításfájlokat. Érdekes például, hogy a fejlesztők gondoltak a multimédia-billentyűzetek tulajdonosaira is: az XF86 által felismert billentyűkhöz az ablakkezelő alapértelmezett programokat rendel (ezeket a keys fájlban nézhetjük meg).
A BlackBoxról, erről az igazi minimalistáról is érdemes néhány szót szólni. Főleg azoknak ajánlom, akiknek öreg csacsi gépen kell dolgozniuk vagy csillogtatni akarják profizmusukat. Legnagyobb előnye ugyanis, hogy döbbenetesen kicsi és gyors. Önmagában a felület nem túlzottan látványos, egy darab tálcát tartalmaz. Az viszont biztos, hogy nem ütközünk olyan gondokba, hogy például az ablakkezelő elnyeli az Alt+F2 vagy az F11 billentyűket, és hogy e billentyűkombinációkat nem tudjuk a programjainkban használni. Ugyanis a BlackBox semmilyen kombinációt nem ismer. Az egész billentyűkezelő részét különválasztották, és külön bbkeys néven tudjuk futtatni. Ennek segítségével viszont egy egyszerű, mégis nagyszerű felületen keresztül állíthatjuk be (például), hogy az Alt+Tab a szokásos módon működjön. Ugyanitt tetszőleges kombinációhoz bármilyen parancsot könnyedén hozzárendelhetünk. Így egyetlen gombnyomásra indulhat egy `xterm`, az Opera vagy bármi más. Remek felület, csak haladóknak!

Szy György

Memóriából futó adatbázis-kezelő rendszerek

A memóriából futó adatbázis-kezelő rendszerek különösen beágyazott gépek esetében lehetnek hasznosak, amelyeknél minden egyes elhagyott folyamattal csökkenhet a termék ára és nőhet a versenyképessége.

Robbanásszerűen növekszik az intelligens, hálózati kapcsolattal rendelkező eszközök iránti érdeklődés. Akár otthon, a saját zsebünkben, akár ipari, távközlési vagy közlekedési berendezésekbe építve egyre nagyobb teljesítményű processzorokkal és egyre kifinomultabb beágyazott operációs rendszerekkel találkozunk. A hasonló eszközökben egyre gyakrabban látható alkalmazások egyik válfaját az adatbázis-kezelő rendszerek (Database Management System – DBMS) családja képezi. Az adatbázisok megszokott lakók az asztali vagy a kiszolgálógépeken, a beágyazott eszközökön azonban még új jövevényeknek számítanak. Mint minden új környezetbe kerülő „élőlénynek", az adatbázisoknak is fejlődniük kell. A DBMS-ek új fajtája a memóriából futó adatbázis-kezelő rendszerek (In-Memory Database System – IMDS) családja, amelyek a faj fejlődésének új szakaszát jelzik.

Vajon miért fordult a beágyazott rendszerek fejlesztőinek érdeklődése az adatbázisok felé? A piaci verseny kikényszeríti, hogy a set-top-box jellegű készülékek, a hálózati kapcsolók és a fogyasztói eszközök egyre okosabbakká váljanak. A növekvő számú szolgáltatás biztosításához az alkalmazásoknak egyre több, egyre összetettebb adatot kell kezelniük. Emiatt számos fejlesztő szembesült azzal, hogy kinőtte saját adatkezelő megoldását, amelynek fenntartása, illetve vele az alkalmazások igényeinek követése egyre nehézkesebbé válik.

Mindemellett az egységesedő, kereskedelmi, a polcról készen leemelhető beágyazott operációs rendszerek terjedése – és ezzel párhuzamosan az egyedi rendszerek visszaszorulása – szintén az adatbázisok elérhetőségét segíti. A széles körben terjedő operációs rendszerek – például a Linux – köré olyan felhasználói kör épül, amely további lendületet ad az adatbázisok és egyéb eszközök kereskedelmi és más jellegű fejlesztésének.

Az eszközök fejlesztői tehát a kereskedelmi adatbázisok felé fordultak, ám a meglévő beágyazott DBMS-rendszerek nem nyújtottak megfelelő megoldást. Az üzleti rendszerek igényeit követve több mint egy évtizede megjelent beágyazott adatbázisok többek között kifinomult gyorstárazásra és a rendellenes leállások utáni helyreállításra is képesek. Egy set-top-boxban vagy következő nemzedékbeli faxkészülékben azonban nem feltétlenül szükséges ilyen tudás, ami sokszor csak a rendelkezésre álló memória és processzoridő kimerülését okozza. Ráadásul a hagyományos adatbázisok az adatokat lemezen tárolják. A lemezes – mechanikus alkatrészek használatával járó – adatforgalom kezelése nagymértékben ronthatja a rendszer teljesítményét. Emiatt a hagyományos adatbázisok a sokszor valós idejű műveleteket végző beágyazott rendszerekben általában túl rossz teljesítményt nyújtanak.

A memóriából futó adatbázisok kifejezetten a beágyazott rendszerek teljesítményre és erőforrásaik korlátozottságára vonatkozó követelményeit figyelembe véve fejlődtek. Mint nevük is sugallja, az IMDS-ek a memóriából futnak, lemezkezelést nem végeznek. Egy IMDS tehát egy hagyományos adatbázis-kezelő lenne, betöltve a memóriába? Jogos a kérdés, hiszen a lemezkezelés elhagyása az új megoldás leginkább szembetűnő jellegzetessége. A Linux eleve képes ramlemezek és memóriában található fájlrendszerek létrehozására. Vajon nem volna-e érdemes egy hagyományos, jól bevált adatbázis-kezelőt, például egy MySQL-t vagy akár egy Oracle-rendszert ilyen lemezekről futtatni?

Az IMDS-ek ennél sokkal több mindenben különböznek beágyazott DBMS-testvéreiktől. Az IMDS-ek egyszerűbbek, mint a hagyományos adatbázis-kezelők. Nemcsak a lemezkezelés képességét nélkülözik, de kevesebb mozgó résszel vagy felhasználói beavatkozást igénylő folyamattal rendelkeznek. Ennek köszönhetően takarékosabban bánnak a memóriával és a processzoridővel, válaszidejük pedig a hagyományos DBMS-ek memóriából történő futtatásával elérhetőnél jóval kedvezőbb.

Ha el kell döntenünk, hogy egy IMDS megfelel-e egy adott tervezet igényeinek, pontosan meg kell értenünk, mely részek kerültek ki a rendszerből vagy módosultak jelentősebb mértékben. A legfontosabb eltéréseket az alábbiakban ismertetem.

Gyorstárazás

A fizikai lemezkezelés által okozott teljesítménycsökkenés miatt gyakorlatilag minden DBMS végez gyorstárazást, hogy az adatbázis utoljára használt részeit a memóriában tartsa. A gyorstárazást vezérlő algoritmus végzi a gyorstárnak a lemezen található adatokkal történő egyeztetését, így a gyorstár tartalma egyezni fog a fizikailag a lemezen található adatbáziséval.

A gyorstár tartalmában végzett keresés, amely azt hivatott eldönteni, vajon a kért adatok a gyorstárban vannak-e vagy sem – az utóbbi esetben kezdeményezni kell a szükséges lap betöltését és hozzáadását a gyorstárhoz a további használat érdekében –, szintén fontos feladat.

Ezek a folyamatok lemezalapú DBMS alkalmazásakor akkor is lejátszódnak, ha a rendszer ramlemezek segítségével fut. A gyorstárazási műveletek eltávolításával az IMDS-adatbázisok lényegesen egyszerűbbekké válnak, kevesebb erőforrást – memóriát és processzoridőt – igényelnek, így nagyobb teljesítményt nyújtanak.

Az adatátadásokból fakadó többletterhelés

Vegyük példaként, hogy milyen műveletek szükségesek ahhoz, hogy egy alkalmazás adatokat olvasson ki egy hagyományos lemezalapú adatbázisból, majd módosítsa, végül visszaírja őket az adatbázisba.

1. Az alkalmazás elkéri az adatokat az adatbázis API-n keresztül az adatbázis motorjától.
2. Az adatbázis motorja utasítja a fájlrendszert, hogy keresse elő az adatokat a fizikai adathordozóról.
3. A fájlrendszer az adatokat bemásolja a saját gyorstárába, illetve a motornak is átadja őket.
4. Az adatbázis egy másolatot helyez el a saját gyorstárában, valamint az adatokat továbbadja az alkalmazásnak.

5. Az alkalmazás módosítja az adatokat, majd az adatbázis API-n keresztül átadja őket az adatbázisnak.
6. Az adatbázis motorja a módosított adatokat bemásolja az adatbázis gyorstárába.
7. Az adatbázis gyorstárának tartalma a fájlrendszerbe időnként kiírásra kerül, ahol elsőként a fájlrendszer gyorstárának frissítése történik meg.
8. Végül megtörténik az adatok kiírása a fizikai adathordozóra.

A fenti lépéseket egy hagyományos adatbázis-kezelőben nem lehet kiiktatni, még akkor sem, ha a teljes feldolgozás a memóriában folyik. Nem szabad elfeledkezni a tranzakciók kezeléséhez szükséges másolatok készítéséről és az adatmozgatásokról sem, amelyekkel a fenti példában nem is foglalkoztunk.

A memóriából futó adatbázisok ezzel szemben kevés vagy semennyi adatmozgatást nem végeznek. Az alkalmazás a saját változóiban ugyan az adatokról készíthet másolatokat, ám lényegében erre sincs szükség. Ehelyett az IMDS olyan mutatókat ad át az alkalmazásnak, amelyek közvetlenül az adatbázisban található adatokra mutatnak, így az alkalmazás közvetlenül az adatokkal dolgozhat. Az adatok védelme így sem marad el, hiszen a mutatók kezelése az adatbázis API-n keresztül történik, amely gondoskodik megfelelő használatukról.

A többszörös adatmozgatás kiiktatása növeli a feldolgozás teljesítményét. A többszörös másolatok elhagyása csökkenti a szükséges memória mennyiségét, az egyszerűbb működés pedig nagyobb megbízhatóságot eredményez.

Tranzakciók feldolgozása

Végzetes leállás, például áramkimaradás esetén a lemezalapú adatbázisok a rendszer újraindításakor a naplófájlok alapján a jóváhagyott tranzakciókat újra lejátsszák, a részlegeseket pedig visszagördítik. A lemezalapú adatbázisokba „beledrótozzák” a tranzakciós naplók fenntartását, illetve az ezek és a gyorstár tartalmának kiírását a lemezre, ha egy-egy tranzakció véget ért.

A memóriából futó adatbázisok is képesek a tranzakciók kezelésére. Ennek érdekében az IMDS megőrzi a módosított vagy törölt objektumok másolatát, illetve listába fogja a tranzakció során hozzáadott adatbázislapokat. Amikor az alkalmazás jóváhagyja a tranzakciót, a korábbi állapotot és az új lapokat tároló memóriarész felszabadul – az eljárás gyors és hatékony. Ha egy tranzakciót meg kell szakítani – mert például a bejövő adatfolyam megszakadt –, a korábbi állapot áll vissza, az új adatokat tároló lapok pedig felszabadulnak.

Végzetes leállás esetén a memóriában tárolt adatbázis elvész – ez az egyik legfontosabb különbség a lemezalapú adatbázisokkal szemben. A készülék bekapcsolása után az IMDS-t újra fel kell tölteni. Nyilvánvaló, hogy nincs értelme tranzakciós naplókat készíteni, amivel újabb összetett, nagymennyiségű memóriát igénylő folyamatot iktattunk ki az IMDS-ből.

Természetesen ezzel a korlátozással nem felelhetünk meg minden környezetben, de a beágyazott rendszerek világában bőségesen találni olyan adattároló és -kezelő alkalmazásokat, amelyek adattárát újraindítás után valós időben gond nélkül újra fel lehet tölteni. Ilyen példa a műsornézegető set-top-box, ami műholdról tölti le az adatokat, vagy egy vezeték nélküli hozzáférési pont, amelyet az útválasztási protokollok hálózati elrendezést felderítő folyamatai révén lehet újra feltölteni adatokkal. A hasonló eszközök fejlesztői boldogan korlátozzák a tranzakciós szolgáltatások kiterjedtségét, ha cserébe növelhetik a rendszer teljesítményét, amely így ráadásul kisebb teljesítményű vassal is beéri.

Természetesen az adatok helyi tárolásáról sem kell teljesen lemondani. Az IMDS segítségével az alkalmazás megnyithat egy folyamot – ez lehet foglalat, csővezeték vagy fájlmutató –, majd utasíthatja az adatbázis-kezelő motorját egy adatbázis-lenyomat beolvasására vagy kiírására a folyamon keresztül. Így készíthető például olyan alapértelmezett lenyomat, amely a rendszer indítása után a működéshez egyfajta kiindulópontot jelent. A folyam másik végén egy program lehet, vagy valamilyen fájl egy tetszőleges fájlrendszen (mágneses, flash- vagy optikai tárolón stb.).

Alkalmazási példa: IP-útválasztók

Hol érdemes IMDS-megoldást választani? A memóriából futó adatbázisok iránti igény számos helyen felmerült, az alábbi környezet talán a leginkább jellemző, az IMDS-megoldás képességeinek kihasználására a leginkább alkalmas: az IP-útválasztó elterjedt, beágyazott operációs rendszert futtató internetes eszköz.

A korszerű IP-útválasztók útválasztási táblakezelő (RTM) programot futtatnak, amely a készülék legfontosabb feladatát, a hálózaton keresztül befutó adatcsomagok következő állomásának meghatározását végzi. Az útválasztási protokollok folyamatosan figyelik a használható útvonalakat és a többi útválasztóeszköz állapotát, majd az eszköz útválasztó tábláját frissítik a megfelelő adatokkal.

Az útválasztó táblák jellemzően az RTM program futásának eredményeként alakulnak ki. Ez a megoldás hozza magával a következő nemzedékbeli útválasztók fejlesztésének egyik legnagyobb kihívását. Ahogy sokasodnak a készülék által ellátott feladatok, az útválasztási tábla kezelése egyre összetettebb munkát jelent. A saját fejlesztésű, az útválasztási adatok kezelését végző programok – az adatbázis-kezelőkkel ellentétben – általában nem képesek az összetett adategyüttesek kezelésére vagy a többszörös hozzáférés biztosítására.

Emellett – ahogyan az alkalmazásba beleépített adatkezelő megoldásoknál is általában – az útválasztó táblák bővíthetősége és megbízhatósága korlátozott. Az adatkezelő eljárások módosítása visszahat a teljes RTM felépítésére, ennek következményeként kellemetlen meglepetések és minőségbiztosítási gondok jelentkezhetnek. Hasonlóan gondot okozó pont a méretezhetőség: az egy-egy feladatra jól használható, saját fejlesztésű adatkezelők általában összeomlanak, ha növekvő terhelés éri őket. Az Internet növekedése eközben gyors előrelépést igényelne az útválasztási megoldások terén, ám az eszközök fejlesztését hátráltatják az elavult megközelítés szerint készített alkalmazások.

A memóriából futó adatbázisok fejlődésével a DBMS-megoldások számos beágyazott rendszerben elérhetővé válnak.

A beágyazott rendszerek fejlesztői számára a már bizonyított adatbázis-kezelő megoldások olyan előnyöket kínálnak, mint hatékony adatelhelyezés és hozzáférési módok, egyszerű és szabványos adatkezelési eljárások, többszörös hozzáférési lehetőség, az adatok épségének védelme, megnövelt rugalmasság és hibatűrés. A DBMS-ek új fajtájának alkalmazásával egyszerűsödhet a beágyazott termékek fejlesztése, miközben nem kell lemondani az összetett alkalmazásokról, a kiváló rendelkezésre állásról és a megbízhatóságról.

Linux Journal 2002. szeptember, 101. szám

Steve Graves
a McObject, az eXtremeDB nevű memóriából futó adatbáziskezelő rendszert fejlesztő cég elnöke és társalapítója. A Raima Corporation elnökeként úttörő szerepet vállalt a DBMS-megoldásoknak a beágyazott rendszereken történő terjesztésében.

Memóriaszivárgások beágyazott rendszereknél

Cal írásában három, az általános memóriakezelési hibák felderítésére alkalmas, könnyen használható eszközről olvashatunk.

A beágyazott rendszerek fejlesztésével kapcsolatos gondok egyike a memóriaszivárgások felderítése. Három programot ajánlok erre a célra. Segítségükkel nem a rendszermag, hanem az alkalmazások memóriaszivárgásai deríthetők fel. Kettő közülük – az `mtrace` és a `dmalloc` – a MontaVista Linux Professional Edition 2.1-es terjesztésnek is részét képezi. A harmadik – a `memwatch` – az Internetről tölthető le (lásd a ***Kapcsolódó címeket***).

A C és C++ nyelvekben programozó fejlesztők dinamikus memória-hozzárendelést használnak. Kellő körültekintés nélkül alkalmazva azonban gondok merülhetnek fel a memóriakezelés környékén, csökkenhet a gép teljesítménye, és megjósolhatatlan lesz az alkalmazás futásának eredménye, vagy összeomolhat a rendszer.

A memóriaszivárgást okozó hibák egy része a lefoglalt területen kívül eső memória olvasásával vagy írásával, illetve a már felszabadított memória újbóli felszabadításával kapcsolatos. Memóriaszivárgás akkor történik, ha a memóriát lefoglaljuk, ám a használat után a felszabadítása nem történik meg; vagy ha a memóriafoglalás mutatóját töröljük, és emiatt a memóriaterületet többé nem lehet használni. A memóriaszivárgás a gyakoribbá váló lapozás miatt csökkenti a gép teljesítményét, illetve idővel a memória elfogyását és az alkalmazás összeomlását okozhatja. A hozzáférési hibák az adatok módosulását okozhatják, ezek miatt a program futásának eredménye megjósolhatatlan lesz, illetve maga a futás is megszakadhat.

Ha egy program kifut a memóriából, akár a Linux rendszermagot is összeomlaszthatja.

A beágyazott alkalmazások tervezését és megvalósítását tehát kellő körültekintéssel kell végezni. Egy alkalmazásnak képesnek kell lennie az összes lehetséges hiba kezelésére, ezeket a hibalehetőségeket pedig gondosan fel kell mérni és el kell hárítani – különösen akkor, ha memóriakezelésről van szó. Gyakran egy alkalmazás többször lefuttatható, mielőtt a soha fel nem szabadított memóriaterületek miatt rejtélyes módon összeomlana, rossz esetben a rendszert is magával rántja.

A hibákat a memóriaszivárgások észlelésére képes programok segítségével kereshetjük meg.

Ezek a programok a `malloc`, a `free` és az egyéb memóriakezelő függvények helyettesítésével működnek. Mindegyik rendelkezik olyan programrésszel, amely elfogja a `malloc` – és egyéb hasonló – függvények hívásait, és minden memóriakéréshez annak követésére alkalmas adatokat fűz. Egyesek memóriavédelemre is képesek, így elfogják a téves memóriahozzáférési kísérleteket.

Vannak olyan memóriaszivárgásokat érzékelő programok, amelyek rendkívül nagy méretűek, és a vizsgált program virtuális memóriaképével dolgoznak. Beágyazott rendszernél azonban az általuk támasztott követelményeket meglehetősen nehéz teljesíteni. Az `mtrace`, a `memwatch` és a `dmalloc` egyszerű programok, ám a legtöbb hibát ezekkel is megtalálhatjuk.

Mindhárom eszközt olyan C-példaprogrammal próbáltuk ki, amely általános memóriakezelési hibákat tartalmazott. A példaprogram a három említett hibakereső program támogatásával történő fordításához szükséges `Makefile` állományokkal együtt a 40. CD Magazin/Szivargas könyvtárában található. A hibakereső programokat különféle típusú célgépekkel is kipróbáltuk. A példaprogram natívan és keresztfordítva is használható.

mtrace

A három említett eszköz közül a legegyszerűbb az `mtrace`. Az `mtrace` a GNU C könyvtár egyik szolgáltatása, segítségével a `malloc/free` hívások egyensúlyának felbillenése észlelhető. Az `mtrace()` függvényhívás által használhatjuk, amely a hívások követését indítja el, és naplózza a `malloc` segítségével lefoglalt, valamint a felszabadított területek címét. Létezik egy szintén `mtrace` névre keresztelt Perl-parancsfájl is, amely a naplófájl kiegyensúlyozatlan hívásait jeleníti meg, és – amennyiben a forráskód hozzáférhető – a kód kérdéses `malloc`-hívást tartalmazó sorát is megjelöli. A program C- és C++-kód ellenőrzésére Linux alatt egyaránt használható. Az `mtrace` leginkább figyelemre méltó tulajdonságra a méretezhetősége. Általános hibakeresésre és modulárisan egyaránt használható.

Az `mtrace` használatának három alapvető eleme van: az ***mcheck.h*** állomány beillesztése a kódba, a `MALLOC_TRACE` környezeti változó beállítása és az `mtrace()` függvény meghívása. Ha a `MALLOC_TRACE` nincs beállítva, az `mtrace()` semmit sem csinál.

Az mtrace kimenete az alábbihoz hasonló:

```
- 0x0804a0f8 Free 13 was never alloc·d
/memory_leak/memory_leaks/mtrace/my_test.c:193
```

(a megjelölt című memóriarész nem volt lefoglalva, ezt követi majd a kipróbált program hibás sorának a száma).

Az üzenet azt jelzi, hogy a memóriát ugyan felszabadítottuk, csakhogy soha nem foglaltuk le, továbbá egy ***Memory not freed*** (Fel nem szabadított memória) rész jelzi azokat a címeket, a kérdéses memória méretét és a hibás kódsorokat, amelyek `malloc`-hívásához nem tartozik memóriafelszabadítás.

memwatch

A `memwatch` nemcsak a `malloc`, illetve `free` használatából fakadó hibákat, de a túlcsordulásokat is felismeri. Túlcsordulás alatt ez esetben azt értjük, amikor az adatok írása – a `malloc` segítségével – lefoglalt memóriarészbe kezdődik, ám a lefoglalt terület határain túl nyúlva történik.

A `memwatch` ugyanakkor nem ismeri fel a már felszabadított területre történő írást, illetve a lefoglalt területen kívül eső memóriarészek olvasását sem.

A `memwatch` központi eleme a ***memwatch.c*** állomány, ez tartalmazza a címellenőrzésekhez használt burkolókat és kódrészeket. A `memwatch` használatához a ***memwatch.h*** állományt a forráskódba be kell illeszteni. A `MEMWATCH` és az `MW_STDIO` változók értékeit a fordításkor parancssorban kell megadni (`-DMEMWATCH` és `-DMW_STDIO`). A fordításkor

természetesen a ***memwatch.c*** fájlra is szükség van.
A ***memwatch.c*** fájlból a fordítás során előálló objektummodult összefűzéskor az alkalmazásba kell építeni. A program futtatásakor bármely rendellenesség észlelése esetén az `STDOUT` kimeneten hibaüzenet jelenik meg. A felfedezett hibákkal kapcsolatos adatokat a ***memwatch.log*** állomány tartalmazza. Minden hibaüzenet megadja a kérdéses programkódot tartalmazó sor számát és a megfelelő állomány nevét.
A ***memwatch.log*** és az `mtrace` által készített naplófájl tartalmát összehasonlítva ugyanazokat a hibákat találtuk. A `memwatch` emellett egy túlcsordulást is észrevett, amelynél a lefoglalt terület elejét és végét jelző memóriacímek módosultak – ezáltal a `memwatch` kiterjedtebb tudása bizonyítást nyert.
A `memwatch` hátránya, hogy nem méretezhető, csak a teljes alkalmazással futtatva használható.

dmalloc

A harmadik eszköz egy könyvtár, amelyet a `malloc`, `realloc`, `calloc`, `free` és egyéb memóriakezelő függvények kiváltására készítettek. Beállításai futási időben is módosíthatók. Képes a memóriaszivárgások követésére, valamint az írásoknál felmerülő túlcsordulások észlelésére. A hibákat fájlnévvel és a kódsor számával jelenti, valamint általános kimutatásokat is készít. A könyvtárat ***Gray Watson*** készítette, átültetései számos, a Linuxtól eltérő operációs rendszerre is létrejöttek.
A csomagot megfelelő beállításokkal a többszálú és C++-programok támogatására is rá lehet venni. Megosztott és statikus könyvtárként is használható. A beállításokat a programfordítás közben lehet megadni. Az eredmény egy olyan könyvtárkészlet, amelyet az alkalmazás összefűzésekor használhatunk fel.
Az ellenőrizendő alkalmazás programkódjába a ***dmalloc.h*** állományt kell beilleszteni. Az ellenőrzés módját, illetve a naplózási beállításokat a könyvtár és a beillesztett állomány használata mellett egy környezeti változó segítségével kell megadni. Az alábbi sort a már említett `dmalloc` példaprogram segítségével végzett ellenőrzéséhez használtuk:

```
export \ DMALLOC_OPTIONS=debug=0x44a40503,
➥inter=1,log=napl fÆjl_neve
```

A beállítások jelentése a következő:
1. a ***log*** adja meg annak a naplófájlnak a nevét, amely a pillanatnyi könyvtárba kerül;
2. az ***inter*** a könyvtár önellenőrzésének gyakorisága;
3. a ***debug*** egy hexadecimális szám, amelynek bitjei a kívánt ellenőrzéseket választják ki.

A fenti példában gyakorlatilag az összes lehetséges hibát naplózzuk. A 40. CD Magazin/Szivargas/debug.txt állományában a választható ellenőrzéseket és a ***debug*** beállításban hozzájuk tartozó biteket soroljuk fel.

Ha a könyvtárat C++-program ellenőrzésére akarjuk használni, egy ***dmalloc.cc*** nevű forrásfájlra is szükség van. Ez a kódrész biztosítja a burkolófüggvényeket a `new` – `malloc` és `delete` – `free` hívásokhoz. A GDB nevű GNU hibakereső és a `dmalloc` együtt is használhatók. A csomag tartalmaz egy fájlt, amelyet a ***.gdbinit*** állományba illesztve a GDB tudomására hozhatjuk a `dmalloc` jelenlétét.
A könyvtár mellé egy `dmalloc` nevű segédprogramot is kapunk, amellyel a `DMALLOC_OPTIONS` változó értékeit módosíthatjuk. Készítettem egy parancsfájlt, amellyel a beállításokat az ellenőrzendő program futtatása előtt is megadhatjuk, így a kipróbálás változatlan körülmények között ismételhető meg.

Jelen cikkben csak az eszköz általános használatáról esik szó, ám leírásában további részleteket, illetve az egyéb szolgáltatások ismertetését is megtaláljuk. A példaprogramot az említett kiegészítések segítségével, a `DMALLOC` használatával futtattuk. Az eredmény akár meglehetősen kiterjedt is lehet – gondoljunk csak a túlcsordulásokra, amelyeknél a mutató túlszalad a lefoglalt területen, és ahol lehetőség van a kérdéses terület tartalmának naplózására. A naplófájl végén kimutatásokat, címeket, blokkméreteket és a megfelelő `free`-hívással nem párosítható `malloc`-hívások kódsorának számát találjuk.
A három említett eszköz különféle módokon támogatja a memóriaszivárgások felderítését és naplózását. Linux-munkaállomáson és keresztfordítással mindet egyaránt kipróbáltuk, és a futtatást különféle célgépeken végeztük. Egy alkalmazás fejlesztésénél a programozók mindhárom eszközt használták. Az `mtrace` segítségével találtak meg egy memóriaszivárgást okozó hibát egy külső fejlesztőtől átvett C++-könyvtárban, amelyben egy kivétel dobása, illetve elkapása okozott súlyosabb gondot. A `dmalloc` segítségével találták meg a memóriaszivárgásokat a linuxos `pthread`-átültetéssel futtatott alkalmazásokban. A `memwatch` roppant hasznosnak bizonyult egy átmeneti tárkezelő rendszer hibáinak felderítésében, amely hibásan végezte önmaga töredezettségmentesítését. Az eszközök kicsik, könnyen használhatók, a hibakeresés végeztével pedig játszi módon eltávolíthatók.
A példaprogram egy ***my_test.c*** nevű fájlból áll. Három külön könyvtárban található: ez a ***README*** és a `Makefile` állomány, illetve egy a próba futtatásához használható parancsfájl (40. CD Magazin/Szivargas). A `dmalloc`-próbához a környezeti változókat beállító parancsfájl mellékelve van. A kipróbálást a Red Hat- és SuSE-Linux terjesztéseken, illetve a MontaVista Linux keresztfordítási környezetében végeztük.

Linux Journal 2002. szeptember, 101. szám

Cal Erickson
jelenleg a MontaVista Software vezető Linux-tanácsadója. Mielőtt csatlakozott volna a MontaVista csapatához, támogatásvezető mérnök volt a Mentor Graphics beágyazott alkalmazások részlegénél. Cal több mint harminc éve dolgozik a számítástechnikai iparágban, tapasztalatát számítógépgyártóknál és felhasználói termékeket fejlesztő cégeknél szerezte. A cal_erickson@mvista.com címen érhető el.

Kapcsolódó címek

A `dmalloc` a ➲ http://dmalloc.com oldalon érhető el. Letölthetők a forráscsomagok, a leírás, a beállító parancsfájlok és egy példaprogram.

Az `mtrace` tekintetében további tájékoztatást az `info libc` parancs kiadásával, majd a *Memory Allocation* (Memóriafoglalás) fejezet *Allocation Debugging* (Memóriafoglalási hibák keresése) című részére ugorva találhatsz.
A `memwatch` a ➲ http://www.linkdata.se/sourcecode.html címen érhető el. Az oldalt John Lindh, a Link Data tulajdonosa tartja fenn.

Icarus Verilog: egy éves késéssel itt a nyílt forrású Verilog

Stephen és Michael a Verilog-tervezéssel kapcsolatos részleteket ismerteti.

Körülbelül 16 hónappal ezelőtt, 2001 februárjában egy ***Stephen Williams***-szel készült beszélgetés keretében egyszer már bemutattuk a Linux Journal oldalain (lásd a ➲ http://www.linuxjournal.com/article/4428-on található cikket) a nyílt forráskódú elektronikus tervezési automatizmusok (Electronic Design Automation, azaz EDA) világát.

Az Icarus Veriloghoz hasonló projektek számára a nyílt forrású fejlesztési módszer számtalan előnnyel jár. Ez a cikk a Verilog segítségével történő tervezésről egy kicsit több módszertani részletet tartalmaz, illetve felfedi az Icarus Verilog fordító alapjainak néhány részletét. Ezenkívül a cikk végén felsorolunk néhány igen kiváló Veriloggal foglalkozó művet, illetve olyan webcímet, ahol jó néhány egyéb nyílt forrású EDA-projektről olvashatunk.

Az Icarus Verilog parancssoros eszköz, amely a Verilogban íródott tervezési forrást fordítja le a célformátumra. Ez a célformátum általában a `vvp` szimulációs motor, a parancssorban azonban más célformátumot is kiválaszthatunk.

Első példánk az ismerős „Szia, Világ!” program lesz:

```
module proba;
initial $display("Szia, vilÆg!");
endmodule
```

Ezt értelemszerűen a következő parancssorozat fordítja le és hajtja végre:

```
% iverilog -oa.vvp hello.v
% vvp a.vvp
Szia, vilÆg!
```

Természetesen a mérnök gyorsan valamilyen érdekesebb példára szeretne továbblépni, ami valamilyen tervezési feladatot old meg. Erre jellemző, egyszerű példa a számlálómodell próbája (lásd az 1. listát: 40. CD Magazin/Verilog könyvtár). A ***szia.v*** és a ***szamlalo1.v*** példákban a fordító egyetlen forrásfájlt kapott meg, amelyet `vvp` formátumú kimeneti fájllá alakított, majd az így elkészített fájlt a `vvp` program hajtotta végre.

A Verilog programokban a modulok azok az objektumtípusok, amelyeket a tervező az eszközök modellezésére hoz létre.

A modulok más modulokat is hívhatnak, példányosíthatnak, illetve saját kódot is tartalmazhatnak, amellyel a modellezendő eszközt írhatják le. A tervező ezután a fő modult hívja meg a teljes modellezett eszköz példányosításához.

A Verilog-fordítók általában úgy állapítják meg, hogy az adott tervben melyek a fő modulok, hogy megvizsgálják, mely modulok nem voltak máshol példányosítva a programozó által adott forrásban. A ***szia.v*** példában mindössze egyetlen `proba` nevű modul volt, ez képezte tehát a fő modult. A ***szamlalo1.v*** programban a `szamlalo` modult a `proba` példányosította, és kizárólag a `proba` volt az egyetlen, amelyet máshol nem példányosítottunk, így a `proba` lett a főmodul.

Az Icarus Verilog használata során a programozók ezt a főmodul-azonosító heurisztikus módszert engedélyezhetik, de a kívánt főmodulokat a `-s` kapcsoló segítségével közvetlen módon is felsorolhatják.

Ahogy növekszik a program, a programozónak egyre inkább több fájlból álló forrás- és programkönyvtárak készítésére lesz szüksége. A programkönyvtárak igen hasznosak a más fejlesztők kiadta piaci eszközöket modellező modulok terjesztésében is. Az Icarus Verilog az önműködő programkönyvtárakat hordozható módon támogatja.

Az önműködő programkönyvtár-formátum ipari szabvány. Ez tulajdonképpen egy olyan Verilog-forrásfájlokat tartalmazó könyvtár, ahol a fájlok egy-egy modult tartalmaznak, és a fájl neve a benne található modulnak felel meg. Például a ***szamlalo.v*** a `szamlalo` modult tartalmazza. A könyvtár helyét az Icarus Verilog-fordítónak parancssorból a `-y` kapcsolóval, vagy parancsfájl használatával adhatjuk meg.

Önműködő programkönyvtárakat használva ***szamlalo1.v*** példánk két részre törik: a könyvtármodulra és a főprogramra. A mostani példában az a logikus, ha a `counter` modult mozgatjuk a ***lib/szamlalo.v***-be és a `proba` modult tartjuk meg a ***szamlalo2.v*** programban. A 2. *listában* megfigyelhetjük, miképpen kell a ***szamlalo2.v***-t két részből álló módon használni.

Az Icarus Verilog a ***szamlalo2.v*** programot először megpróbálja lefordítani. Amikor a `szamlalo` modul példányosításához ér, észreveszi, hogy nincsen `szamlalo` modulmeghatározás, ezért végignézi az általa ismert (`-y` kapcsolóval megadott) könyvtárakat, és megtalálja a ***lib/szamlalo.v*** fájlt. Értelmezi az új Verilog-forrást, menti a megtalált modulmeghatározásokat, és folytatja az eredeti forrás fordítását. A programkönyvtárban való keresés rekurzív, így a könyvtármodulok más könyvtármodulokat is példányosíthatnak, miközben a fordító a modulmeghatározásokat folyamatosan gyűjti, míg a program el nem készül.

Az Icarus Verilog további kényelmi szolgáltatása, hogy parancsfájlokat képes feldolgozni. A parancsfájlok olyan szöveges állományok, amelyek fájlneveket, útvonalmegadásokat és más fordítási irányelveket tartalmaznak. A ***szamlalo2.v*** példánkhoz ilyesféle parancsfájlt tudnánk készíteni:

```
szamlalo2.txt:
  # ez egy sajÆt k nyvtÆr
  -y lib
```

2. lista A szamlalo2.v példa

```
\begin{verbatim}
module proba;

/* Egyszer pulzÆl  reset kØsz tØse. */
   reg reset = 0;
   initial begin
      # 17 reset = 1;
      # 11 reset = 0;
      # 29 reset = 1;
      # 11 reset = 0;
      # 100 $stop;
   end

/* hagyomÆnyos pulzÆl   ra. */
   reg clk;
   always #5 clk = !clk;
   wire [7:0] value;
   szamlalo c1 (value, clk, reset);


   initial $monitor("At time %t, value
                    ➥= %h (%0d)",
                    ➥$time, value, value);
endmodule // test

\end{verbatim}
\caption{Example Verilog simulation file
  \texttt{szamlalo2.v}}\label{fig2}

    % iverilog -o a.vvp -ylib szamlalo2.v
    % vvp a.vvp
    [...]
```

```
  # az ehhez a beÆll tÆshoz tartoz  projekt-
  #  forrÆsfÆjlok
  szamlalo2.v
```

Ezután a programot a következőképpen futtathatnánk:

```
% iverilog -c szamlalo2.txt -o a.vvp
% vvp a.vvp
```

Egy ilyen kis program esetében a parancsfájl használatának nincs túl sok értelme, de ahogyan a terv egyre nagyobbá válik (forrásfájlok százai sem ritkák), a parancsfájl értéke igencsak megnő.
Az Icarus Verilog rendelkezik egy más Verilog-fordítókban fel nem lelhető különleges képességgel is: támogatja a betölthető cél-API-kat. A betölthető modulokhoz készült C API-t a fordító akkor hívhatja meg, amikor a kimenetet új formátumban kell előállítania. A vvp kódelőállító maga is ilyen betölthető célmodul, amely a(z alapértelmezett) vvp-szimuláció kérése esetén indul el. Létezik egy `null` kódelőállító és egy `fpga`-kódelőállító is.
A betölthető cél-API a C-programozók számára az Icarus Verilog-csomagokkal együtt érkező és települő ***ivl_target.h header*** fájlon keresztül érhető el. Ezáltal a C-programozók új kimenet-előállítókat írhatnak az Icarus Verilog fordítóhoz.

A vvp futtatható állomány szintén támogatja a szabványos Verilog VPI-csatolófelület egy részét. Itt egy olyan futásidejű C API-ról van szó, amely a programozóknak lehetővé teszi, hogy a Verilog-forrás által meghívható új rendszerfeladatokat illesszenek be. Ilyen rendszerfeladatokra példa a `$stop` és a `$monitor` utasítás a ***szamlalo2.v*** példa forrásfájljában. Minden szabványos magrendszerfeladat Icarus Verilogban íródott, a VPI API segítségével.

Linux Journal 2002. július, 99. szám

Stephen Williams
egy szép napon eltévedt egyetemének számítógéptermében. Bár elektromérnök-hallgató volt, informatikusként (computer science) végzett. Tíz évvel később, miközben számos programfejlesztésben részt vett – különös tekintettel a nagy és erősen osztott rendszerekre –, egyszer csak azon kapta magát, hogy nagysebességű digitális kamerák vezetékeibe gabalyodik. Azóta e tapasztalatokat ötvözve eszközmeghajtókon, beágyazott rendszereken és EDA-eszközökön dolgozik.

Michael Baxter
kilencéves kora óta dolgozik a számítógépiparban, amióta magával ragadta az 1969-es 2001-vízió, az Ürodüsszeia. Tapasztalt számítógép-, rendszer-, áramkör- és FPGA-tervező. Tíz USA-szabadalmat jegyez számítógépek és áramkörök tervezésével kapcsolatban, valamint további ötnek társfeltalálója. Szeret túrázni, érdekli az amatőr rádiózás és a Lisp nyelven való programozás.

Hasznos Verilog-források

IEEE Std. 1364-1995 (ISBN 1-55937-727-5)
➲ http://standards.ieee.org
IEEE Std. 1364-2001 (ISBN 0-7981-2806-6)
➲ http://standards.ieee.org
Verilog 2001: A guide to the New Features of the Verilog Hardware Description Language by *Stuart Sutherland*. Kluwer Academic Publishers, 2001. ISBN 0-7923-7568-8.
The Verilog PLI Handbook: A User's Guide and Comprehensive Reference on the VERILOG Programming Language Interface by *Stuart Sutherland*. Kluwer Academic Publishers, 1999. ISBN 0-7923-8489-x.
Verilog Quickstart, Second Edition by *James M. Lee*. Kluwer Academic Publishers, 1997. ISBN 0-7923-8515-2.

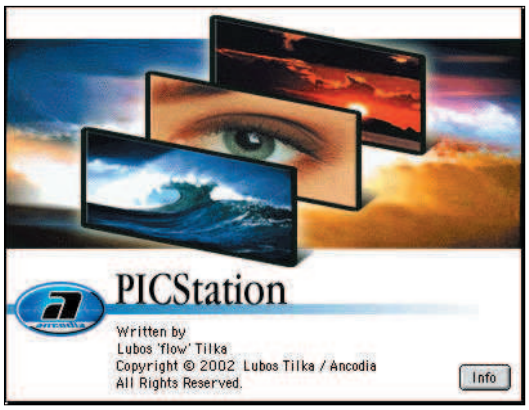
Néhány nyílt forrású EDA-honlap

gEDA ➲ http://www.geda.seul.org
Icarus Verilog ➲ http://icarus.com/eda/verilog
Open Collector ➲ http://www.opencollector.org
OPENCORES.ORG ➲ http://www.opencores.org

Programok telepítése almára

Nézzük, milyen programokat is tudunk a jó öreg Mac gépre telepíteni!

Először is nézzünk körül a házunk táján: ha van Macintosh gépünk (ugye, van? És a ROM-ot nem törvénytelenül használjuk!), akkor programok is tartoznak hozzá – hajlékonylemezen, CD-n vagy bármilyen adathordozón. Válasszuk ki az áldozatot, és kezdjünk neki a telepítésnek. Előtte azonban nézzük meg, milyen fájlformátumokkal lesz dolgunk: ***.hqx***, ***.bin*** és ***.sit***. Én abba a csapdába estem, hogy a pillanatnyilag legfrisebb Stuffit Expander csomagkezelő nem kezelte megfelelően a csomagokat, ahhoz azonban, hogy egy újabb változatot telepítsek, szintén működnie kellett volna, de nem tette! Mit csinál az ember a 22-es csapdájába esve? Feltalálja magát! Kerestem egy programot Linux alá, amivel ki tudtam bontani a csomagokat (ezen a címen megtalálható: ➲ http://www.stuffit.com/stuffit/linux/). Ezek után már rendelkeztem egy „normális” Stuffit Expanderrel, úgyhogy folytathattam ámokfutásomat az almák között.



Programok beszerzése

Természetesen akkor van a legkönnyebb dolgunk, ha meglévő programjainkat telepítjük az emulátorban. Némelyik program sokkal gyorsabban fog futni, mint azt jó öreg gépünkön megszoktuk (ha még emlékszünk rá, milyen sebességgel is futottak azon). A programok tárháza szinte végtelen, és biztosan mindenki tudja, hogy ezeken a gépeken is futottak a profi DTP-s programok korábbi változatai, mint például a QuarkExpress, az Adobe-programok. Ezek ingyenesen sajnos nem érhetők el, bár igazándiból nem értem, hogy miért nem lehet ezek régebbi változatait – akárcsak a MacOS 7.5.3-ast, illetve a 7.5.5-ös frissítést – szabadon letölthetővé tenni.
Az ingyenesen letölthető programok weblapjai a ***Kapcsolódó címek*** alatt találhatóak. Viszonylag sok program fellelhető az Interneten (bár többnek nem sok értelme van azon kívül, hogy fut). Első nagy kedvencem a ***Car Care For The Beginer 1.1***-es program (kezdők autóápolási kézikönyve) Ez számos érdekességgel szolgál az újdonsült autótulajdonosoknak. Megtudhatjuk, miből áll az autó, miként működnek a különböző részei, illetve kapunk egy áttekintő hibaelhárítási útmutatót is (igaz, csak angolul). A másik nagyon kedves program a ***Paragon Poker Suite*** kártyajáték. Letölthető változatának sajnos nem minden része használható, de 10 dollárért hozzájuthatunk a program többi részéhez vezető kulcsokhoz. Ez a program beszél, méghozzá úgy, hogy nyomon követi a játékosokat, és mindent hangosan is közöl: ki emelt tétet, ki dobta be a lapokat, ki hány lapot cserélt. Az elején jópofa ez a hangos játék, huzamosabb ideig játszva azonban szerintem mindenki letekeri a hangerőt. A következő játék a népszerű Mahjongg egyik képviselője, ami ***A Szelek Játéka*** (The Game Of The Winds) nevet viseli. Kellemes, csak egy kicsit szemrontó játék.
A ***Musashi 3.4.1*** remek levelezőprogram. Támogatja a többfelhasználós módot, több postafiókot kezel egyszerre, úgyhogy aki emulátorból szeretne levelezni, hát hajrá! A SonoSoft oldaláról több programot is letölthetünk.
A grafikai programok listáját a ***PICStationnel*** kezdem, ami kitűnő képnézegető alkalmazás: képes megjeleníteni a Photoshop, JPEG, TIFF, PICT, SGI, TARGA, BMP, PNG formátumokat. Olyan alapvető képfeldolgozási műveletekre is használható, mint az átméretezés, a forgatás, a kontrasztállítás, de kötegelt feldolgozásra is kiváló.
A ***PrintToPDF*** programot az állományok PDF fájlba történő nyomtatásához használhatjuk, így könnyedén készíthetünk Acrobat Reader által olvasható fájlokat.
Kellemes játékszer a ***World Clock Deluxe:*** akár 24 különböző órát is kirakhatunk az asztalunkra, így követve nyomon a világban az idő múlását.
Az alábbi programok, azt hiszem, nem igényelnek magyarázatot:

- Apple Quicktime 4.0
- Real Player 5.0
- Netscape Communicator 4.08

Mindezeket látva úgy gondolom, immár senkinek sem kell régi számítógépe ellátatlanságáról panaszkodnia. Miközben ezt a cikket írtam, azon törtem a fejemet, vajon mi lehet az az erő, ami az embereket a még újabb, még nagyobb gépek és programok felé hajtja? Erre akkor kaptam meg a választ, amikor odaültem egy Macintosh Quadra 605-ös gép elé és megpróbáltam olyan sebességgel dolgozni, mint a 1,2 GHz-es laptopomon MacOS 7.5.5-öst emulálva. Nos kérem szépen, az eredmény és a látvány meglepő volt, úgy éreztem magam, mintha zselében számítógépeznék.

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

Kapcsolódó címek

➲ http://guide.apple.com/
➲ http://www.macoszone.com/
➲ http://www.ancodia.com/software/picstation/

PHP PEAR (2. rész)

Ezúttal a PHP PEAR levélkezelő osztályaival ismerkedünk meg, amelyekkel egyszerűvé válik a csatolt fájlokat és színes képeket, szövegeket tartalmazó üzenetek kezelése.

Sorozatunk előző részében – mely az adatbázisok egységesített kezeléséről szólt – már találkoztunk a PEAR-rel. Arról azonban nem beszéltem, hogy a PEAR adatbázis-kezelési képességein kívül még megannyi hasznos szolgáltatást tartogat a számunkra. Számos mindennapos feladatra – legyen szó akár adatbázis- vagy hibakezelésről, hálózatokról, akár XML-ről – a PEAR kész megoldásokkal szolgál.
Ugye a `mail()` függvényt te is sokszor használtad már? Valószínűleg. Bizonyára azt is tapasztaltad, hogy ezt a függvényt nem éppen arra tervezték, hogy tetszetős HTML-leveleket küldjünk vele. Sőt, egy egyszerű csatolt fájl elküldése is lehetetlen vállalkozásnak tűnhet a megfelelő segédeszközök nélkül. Ilyenkor lehet hasznunkra PEAR MIME levélkezelésre szolgáló bővítménye.

A MIME

A MIME a Multipurpose Internet Mail Extensions rövidítése. A MIME révén válik lehetővé – nemcsak PHP-ből, hanem bármely szabványos levelezőügyfélből –, hogy csatolt fájlokat, vagy több részből álló üzeneteket küldjünk. De a MIME felelős a levelek tartalmának helyes megjelöléséért is, és nélküle még egy ékezetes levél megírása is nehézségekbe ütközne.
A MIME a szabványos RFC 822-es elektronikus levél kiterjesztéseként jött létre. Az RFC 822 maga a szabvány, ebben van meghatározva mindaz, amit ma az Interneten a levél fogalmával azonosítunk. A MIME pedig egy bővítmény a szabványhoz, amelynek ugyanúgy megvannak a maga RFC-i.

Az első lépés

Mielőtt bárminek nekikezdenénk, telepítsük a PEAR levélkezelésre szakosodott Mail-bővítményét a következőképpen:

```
# pear install http://pear.php.net/get/Mail
```

Ahhoz, hogy ez a parancs sikeresen lefusson, először a PEAR telepítésére lesz szükség, ahogyan ezt az előző cikkben már taglaltam. Ha elakadnál, lapozd fel a Linuxvilág előző számát vagy látogass el a ➲ http://pear.php.net/ oldalra, ahol minden tudnivalót megtalálsz.
Ezt követően pedig telepítsük a `Mail_mime`, `Mail_mimePart` és `Mail_RFC822` bővítményeket is.

Egyszerű levélküldés

Ha nem vagyunk nagy igényűek, és csupán egy egyszerű, ékezetektől mentes levelet szeretnénk küldeni, a `Mail` osztályt is használhatjuk, ami a PHP `mail()` függvényétől talán csak annyiban különbözik, hogy a parancsfájlból meghatározhatunk bizonyos beállításokat. Például megadhatjuk, hogy a program a Sendmailt használja a levél elküldésére vagy – mint ahogyan az *1. listán* szereplő példánkból is kitűnik – egy külső SMTP-kiszolgálót használjunk erre a célra. Mielőtt kipróbálnánk a programot, ne felejtsük el telepíteni a `PEAR` `Net_Socket` és `Net_SMTP` bővítményeit.

A példában látható kód viszonylag egyszerű: a `$params['host']` változóban adjuk meg a küldéshez használni kívánt SMTP-kiszolgálót (ami jó esetben ***mail.szolgáltatónkneve.hu*** formájú), a `$headers` tömbben pedig a levél fejlécének elemeit határozzuk meg.
A `$headers['To']` és a `$recipients` változó annyiban különbözik, hogy a `$recipients` változóban a levél tényleges címzettjeit kell megadni, míg a `$headers['To']`-ban csak a levél fejlécében látható címzett nevét. Ez utóbbi tartalma a levél küldése szempontjából közömbös, ennek ellenére ezt sem árt helyesen kitölteni.
A `Mail::factory()` tagfüggvény egy `mail` objektummal tér vissza, amelyet a `send()` tagfüggvénnyel bocsáthatunk útjára.

1. lista Egyszerű levélküldés az smtpsend.php segítségével

```
<?php
include('Mail.php');
$recipients = 'ggabor@sopron.hu';
$headers['From'] = 'valaki@example.com';
$headers['To'] = 'ggabor@sopron.hu';
$headers['Subject'] = 'Teszt uzenet';
$body = 'Ez egy probalevel.';
$params['host'] = 'mail.sopron.hu';

// A Mail osztÆly factory() tagf ggvØnyØvel
lØtrehozunk egy mail objektumot
$email =& Mail::factory('smtp', $params);
$email->send($recipients, $headers, $body);
?>
```

Az ilyen módon küldött levelek semmiképpen nem kódolódnak, egyszerű 7 vagy 8 bites levélként postázódnak, ezért ha a levélbe ékezetet is írunk, fennállhat a veszély, hogy a címzett esetleg képtelen lesz elolvasni a neki szánt üzenetet. Emiatt csak akkor küldjünk levelet ilyen módon, ha biztosan nem használunk ékezeteket.

A színfalak mögött

Ahhoz, hogy MIME-leveleket küldjünk, nem árt, ha egy kicsit belepillantunk, mit is takar ez a négy betű valójában. Erre azért van szükség, hogy később, amikor MIME-leveleket állítunk össze, megértsük az eljárás részleteit. Nem kell megijedni, nem lesz ez olyan bonyolult. Nem is szaporítom tovább a szót, nézzük!
Minden MIME-levél fejléce tartalmaz bizonyos elemeket, amelyek a levelet megjelenítő programnak szólnak.
A MIME a fejléceit nemcsak a szabványos üzenet fejlécéhez adhatja hozzá, hanem a saját részüzeneteihez is. Részüzenet alatt a többrészes – vagyis a HTML-szöveget vagy mellékletet tartalmazó – üzeneteknél a levélben található

2. lista Példa MIME-levélre

```
From: "Gludovatz Gabor" <ggabor@sopron.hu>
To: Valaki <valaki@example.com>
Subject: Egy egyszeru MIME level
Date: Wed, 20 Jun 1999 17:18:47 +0100
Message-ID: <000000123@localhost>
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="Linuxvilag-123";
Content-Transfer-Encoding: 7bit

Ha az uzenetnek ez a resze lathato, akkor
a levelet olyan levelezoprogramban olvassak,
mely nem kezeli a MIME uzeneteket.

 Linuxvilag-1233
Content-Type: text/plain; charset="iso-8859-2"
Content-Transfer-Encoding: quoted-printable

Hell=F3 Valaki! Ez itt egy MIME e-mail!

  Gabor

 Linuxvilag-1233
Content-Type: image/jpeg; name="kep.jpg";
Content-Transfer-Encoding: base64
Content-Disposition: attachment
<a kep.jpg tartalma base64-gyel k dolva>

 Linuxvilag-123 --
```

3. lista A mimemail.php – HTML-t is tartalmazó üzenet küldése

```
<?php
include('Mail.php');
include('Mail/mime.php');

$text = 'A levØl sz veges vÆltozata.';
$html = '<html><body>A levØl <b>HTML</b>-es
    ➥vÆltozata.</body></html>';
$file = '/tmp/logo.jpg';
$crlf = "\r\n";
$hdrs = array(
            'From'    => 'ggabor@sopron.hu',
            'Subject' => 'Pr ba  zenet'
            );

$mime = new Mail_mime($crlf);

$mime->setTXTBody($text);
$mime->setHTMLBody($html);
$mime->addAttachment($file, 'image/jpeg');

$body = $mime->get();
$hdrs = $mime->headers($hdrs);

$params["host"] = "mail.sopron.hu";
$mail =& Mail::factory('smtp', $params);
$mail->send('ggabor@sopron.hu', $hdrs,
$body);
?>
```

HTML-részt vagy fájlmellékletet értjük.
A MIME fejléceinek többsége mind az elsődleges fejlécben, mind a részüzenet fejlécében megjelenhet, kivétel ez alól a ***MIME-Version:*** mező, mely az adott levél létrehozásához használt MIME-változatot tartalmazza.
A ***Content-Type:*** mező az üzenet tartalmára utal. Lehetséges értékei például ***text/plain***, ***text/html***, ***image/jpeg***, amelyek a levél vagy a részüzenet tartalmára utalnak. Ha a ***Content-Type:***-ot az elsődleges fejlécben találjuk meg, az értéke valamilyen ***multipart/**** érték is lehet, például ***multipart/mixed*** vagy ***multipart/alternative***.
A ***Content-Transfer-Encoding:*** az adott üzenetrészben található tartalom kódolására utal. Mivel alapértelmezés szerint a levelekben csak 7-bites adatot küldhetünk, olyan módszereket kellett kitalálni, amelyekkel lehetővé válik 8-bites ékezetes karakterek, illetve fájlok küldése. Ékezetes tartalom küldéséhez a „quoted-printable” kódolást szokás használni, ha azonban a levél nem tartalmaz ékezetes karaktereket, elegendő a 7 bit is. Bináris fájlok esetén a `base64` kódolás használatos, ami a fájlt úgy alakítja át 7-bitesre, hogy az később kibontható, felhasználható legyen. A 8 bit és `binary` kódolási típusok használata kevésbé elterjedt (emiatt a működésük sem mindenütt biztosított).
A ***Content-Disposition:*** egy egyelőre még kísérleti állapotban lévő fejléc. Segítségével a levélolvasó program egyszerűen eldöntheti, hogy mellékletről vagy a levél tartalmához kapcsolódó anyagról van-e szó.
MIME-levélre láthatunk példát *2. listánkon*.

4. lista Az rfc822.php levélcím ellenőrzése

```
<?php
$cim = 'Egy csoport: "GludovÆtz GÆbor"
<ggabor@sopron.hu>;, valaki@example.com
(Egy megjegyzØs)';
$rfc822 = new Mail_RFC822($address,
'example.com', true);

$cimek = $rfc822->parseAddressList();
print_r($cimek);
?>
```

HTML-üzenetek küldése

Amennyiben levelünket HTML-üzenetként szeretnénk elküldeni, a `Mail_mime` osztályhoz fordulhatunk segítségért. Az ezzel a bővítménnyel létrehozott üzenetekben található ékezetek és fájlok a szabványnak megfelelően kódolódnak, így nyugodtan bármilyen szöveget írhatunk, biztosak lehetünk benne, hogy a címzett el fogja tudni olvasni a levelet – már amennyiben a levelezőprogramja támogatja a MIME-t (a ma használatos levelezőprogramok szinte kivétel nélkül támogatják). Programunk forráskódja a *3. listában* olvasható. A levél közvetlen elküldéséhez ezúttal is a PEAR Mailt hívtuk segítségül, a levél tartalmát azonban a `Mail_mime` osztállyal állítjuk elő. Az osztály használata meglehetősen egyszerű,

beállítjuk levelünk szöveges tartalmát, beállítjuk a HTML-tartalmat, csatolunk egy képet, és már küldhetjük is a levelet, anélkül, hogy bele kellene bonyolódnunk a részletekbe.
HTML-ben írt levél esetében a levélhez mellékelni kell az üzenet egyszerű szöveges példányát is, arra az esetre, ha a címzett levelezőprogramja nem képes html-es üzenetek megjelenítésére. A szöveges és html-es tartalom közül a címzett levelezőprogramja azt a részt fogja kiválasztani, amelyek számára a legmegfelelőbb.
Az így létrejött levél ***multipart/mixed*** lesz, amelynek második részüzenete ***image/jpeg*** típusú, ez tartalmazza a mellékelt képet; ezzel szemben első részüzenete ugyancsak `multipart` típusú, mégpedig ***multipart/alternative***.
A ***multipart/alternative*** rész egy beágyazott MIME-üzenetet takar további részüzenetekkel, de a ***multipart/mixed***-hez képest azzal a különbséggel, hogy a ***multipart/alternative*** részben mindegyik melléklet ugyanazt a tartalmat hordozza, és a levelezőprogram a ***multipart/alternative***-ból választja ki azt a részt, ami számára a legmegfelelőbb. Ez annyit tesz, hogyha egy levelezőügyfél nem tud HTML-t, a ***text/plain*** típusút választja; ezzel szemben egy mindenféle extrával felszerelt levelezőprogram a ***text/html*** típust fogja kiválasztani.

Levélcímek ellenőrzése

A PEAR levelekkel kapcsolatos fejezetében található még egy hasznos osztály, amivel az oldalainkon megadott levélcímeket ellenőrizhetjük, hogy valóban szabályosan adták-e meg őket. Ez az osztály azért hasznos, mert egy levélcím ellenőrzése sokszor nagyon bonyolult feladat, és ha nem akarunk hamisnak tűnő, ám mégis szabályos címeket is eldobni, ezt az osztályt ajánlatos segítségül hívni. Az így ellenőrzött levélcímek biztosan megfelelnek az RFC 822-ben meghatározott szabványnak.
A bővítmény használatára a *4. listában* látható egy példa.

Összegzés

A levélküldés bizony néha bonyolultabb, mint elsőre feltételeztük volna. Ilyenkor jönnek jól a PEAR levélküldésre szakosodott osztályai, amelyekkel bármilyen típusú levelet könnyedén elküldhetünk. A PEAR tehát ezúttal is jó segítőtársnak bizonyult.

Gludovátz Gábor (ggabor@sopron.hu) egy soproni cég Linux-rendszerekkel foglalkozó rendszergazdája. Kedvenc időtöltése a programozás, és a Linux lelkivilágában való kutakodás. Ha ideje engedi, szívesen hódol szenvedélyének és bringáján a környező erdőket járja. Honlapja a ➲ http://www.sopron.hu/~ggabor/ címen érhető el.

Kapcsolódó címek

PEAR ➲ http://pear.php.net/
PEAR-leírás ➲ http:// pear.php.net/manual/en/
A MIME működése
➲ http://www.zend.com/zendspotlight/sendmimemailpart1.php
➲ http://www.phpbuilder.com/columns/0kartic20000807.php3
RFC-k keresése ➲ http://rfc.sunsite.dk/

A Linux nem csak operációs rendszer: egyesek szerint életérzés, mások egyenesen vallásként élik meg. Az igazi Linux-felhasználó szabadnak érezheti magát: a szabad program továbbadható másoknak, korlátlan számban másolható és felhasználható. Kezünket nem kötik ipari titokként féltve őrzött kódok, és a hozzáférésen túl a programok módosítása is lehetséges, ami azt is jelenti, hogy előttünk százak és ezrek már éltek e lehetőséggel, kijavítva számos hibát. A Linux a legtöbb élőmunkát tartalmazó operációs rendszer, emellett terhelhető, igen megbízhatóan és hatékonyan használja a számítógép nyújtotta erőforrásokat – és nem utolsósorban használata komoly költségmegtakarítást jelent. *Pere László* kötete ugyanakkor nem hallgat kedvenc rendszerünk hibáiról, hiányosságairól sem. Természetesen hiteltelenné válik az a könyv, amelyben nincs lelkesedés, így a „térítésnek" is fenntart egy kicsiny fejezetet, a könyv többi része azonban a száraz tényeket mutatja be, körüljárva minden területet, amely a kezdő és haladó Linux-felhasználó számára fontos lehet.

ISBN: 963 9301 37 x
Ár: 2660 Ft, 249 oldal
Felhasználói szint: kezdő

A főbb témakörök:

- A Linux felépítése
- Bejelentkezés a rendszerbe, a felhasználó azonosítása
- A konzol használata
- A könyvtárbejegyzések
- A héj használata: parancsok, utasításcsövek, a héj testreszabása és programozása
- Állománykezelés: hivatkozások, meghajtók beillesztése a rendszerbe, tömörítés, biztonsági mentések
- Szabályos kifejezések és szűrők
- Feladatvezérlés: a folyamatok irányítása és figyelése
- Kapcsolattartás és levelezés
- Szövegszerkesztés: a vi és a Midnight Commander használata, külön táblázatokkal, amelyek a billentyűparancsokat tartalmazzák
- Héjprogramozás
- Számítógép-hálózatok és a Világháló
- Grafikus felületek: az X Window rendszer és az ablakkezelők
- Kiadványszerkesztés: részletes LaTeX-ismertető, az alapvető dokumentumformázási parancsoktól a különleges karakterek és matematikai képletek használatáig, illetve az utómunkálatokig, részletes táblázatokkal.

GPRS Linux alól is

Elsőként tisztázzuk, milyen sebességet várunk el az új GPRS csomagkapcsolt adatátviteltől. Filmeket, terjesztéseket akarunk töltögetni?

A mennyiben a válasz igen, bele se kezdjünk egy GPRS-hálózat felállításába. Rengeteg írás olvasható a Neten a Vodafone új, átalánydíjas internetszolgáltatásáról. A cikkek többsége igen lehangolóan számol be arról, milyen hihetetlenül lassú a hálózat stb. Nos nézzük, eddig mire is használtuk a mobil Internetet (nem azonos a WAP-pal!).
Ha rendszerfelügyelőként vidékre kellett utaznunk és onnan bejelentkeznünk, csak felcsörögtük a jó kis megbízható 9600-ast, és SSH vagy VPN segítségével belépett a megfelelő helyre. Kiböjtöltük a szédületes sebességgel felénk áramló adatokat, és tettük, amit kell. Ha a legújabb mozifilmeket le akartuk tölteni a megjelenésük előtt 2–3 hónappal, bizony a cég bérelt vonalát vagy az otthoni kábelnetet használtuk. Most se legyen ez másképpen. Ha a GPRS-hálózatot úgy fogjuk fel, mint egy jó dolgot, ami nem azért van, hogy warezoljunk, akkor rendszerfelügyeletre és a leveleink letöltésére, esetleg egy-két weboldal megnézegetésére (csak a bátrabbaknak) tökéletes megoldás, hiszen nem az eltöltött idő után fizetünk, hanem a forgalmazott bitmennyiség után. Nézzünk egy példát! Béla vezető rendszermérnök, és Szolnokra utazik előadást tartani. Természetesen viszi magával a hordozható számítógépét vagy legalább egy tenyérgépet. Mikor máskor, mint az éjszaka kellős közepén megáll az egyik szolgáltatás, Béla bejelentkezik GPRS-hálózaton és 2,5 órán keresztül üldözi a bogarakat (Debug), mire megtalálja a hibát. A hiba természetesen abból eredt, hogy elfelejtette kikapcsolni az adatbázismotor részletes naplózás lehetőségét, így két nap alatt 3 GB-nyi naplófájl gyűlt össze. A hely elfogyott és a szolgáltatások hely hiányában szépen sorban leálltak. Béla a rendszerfelügyelet ideje alatt leginkább a `less`, `vi`, `rm` parancsokat használta egy SSH-ablakban. Tehát a kiszolgálón dolgozott, a klasszikus értelemben véve, fájlokat se le, se föl nem töltött. Cége a 2,5 óra GPRS-forgalmáért megközelítőleg 160 forintot fizetett, mivel összes forgalma nem érte el a 200 KB-ot. Ebből a példából is jól látszik, hogy mikor előnyös a GPRS használata. A Vodafone hálózatában ezt a szolgáltatást átalánydíjjal kínálják, a hálózatról csak annyit, hogy most az ingyenes, bevezető szakaszban nagyon túlterhelt, de 9600-as sebességgel szinte mindig megy. A Westel és a Pannon GSM hálózata lehet, hogy jobb ennél, hiszen ott a forgalmazott adat után keményen fizetni kell, így valószínűleg nincs túlterhelve.

Készülékkínálat

Mielőtt rátérnénk, hogyan állítsuk be a GPRS-t Linux alatt, nézzük, milyen választási lehetőségeink vannak a mobiltelefon-kínálatból. A jelenleg piacon lévő GPRS-t támogató készülékek 99 százaléka Linux alatt működik, hiszen szabványos modem található bennük. Számos kedvezőtlen próbaeredményt lehet olvasni szinte az összes készülékről, hiszen ezek többségét nem kifejezetten végberendezésnek (modemnek) tervezték. Általános tanácsként mindenkinek azt tudom javasolni, hogy olyan készüléket vegyen, amelyiket egyszerre lehet tölteni és közben adatot forgalmazni rajta. Ehhez az szükséges, hogy az adatkábel a töltő elől ne foglalja el a nyílást, vagy az adatátviteli kábelhez a töltőt is hozzá lehessen illeszteni. Léteznek olyan megoldások is, amikor egy további átalakító segítségével a két kábel egyszerre is beköthető. Ilyen például a Siemens ME45-ös készülék, amelynél egy plusz 4000 forintos kábellel megoldható, hogy a töltő és az adatkábel egyszerre legyen a készülékhez csatlakoztatva. További szempont, hogy tudja-e a készülék menet közben tölteni magát? Például az Ericsson R600-as telefon sajnos ezt nem tudja. A szerviz állítása szerint ez a telefon hatékony töltésének érdekében nem oldható meg. Gyakori gondként szokott jelentkezni, hogyha a kapcsolat megszakad, vagy a `pppd`, vagy a telefon belső vezérlőprogramja „beragad”. A `pppd`-t még csak ki lehet lőni terminálból, de a telefont nagyon idegesítő ki-bekapcsolgatni. Erre jelent megoldás, ha az adatkábelt USB-kapun keresztül illesztjük a géphez, mert ilyenkor lehetőségünk nyílik, hogy az eszközvezérlőt modulként fordítsuk a rendszermagba. Így ha a kapcsolat beragad, nem kell le-lehúzogatni a kábelt a telefonról, egész egyszerűen csak az `rmmode` paranccsal visszatöltjük az USB-vezérlőt, és újra működésbe hozzuk. Ez a gép szempontjából csaknem a fizikai lehúzással egyenértékű. Ennél egyszerűbb, ha a telefont IRDA-vezérlőn keresztül kapcsoljuk a géphez, hiszen a csatlakozó ilyenkor szabad, tehát tölteni mindig lehet, és az infraeszközmodulban tökéletesen működik. Egy okos figyelőprogrammal bármikor megállapíthatjuk, mikor állt meg az adatforgalom, és újra tudjuk indítani. Néhány telefonnál az IRDA állandó működésre kapcsolható, vannak azonban olyan modellek, amelyeknél mindig kézzel kell újraindítani.
Mivel mindegyik telefon viszonylag drága, egy járható és törvényes trükköt javasolnék.
A Vodafone-nál kapható az ezüst P7389-es Motorola Timeport, ami egy viszonylag jó telefon. Sajnos azonban nem tudja a GPRS-t. A telefont előre fizetett előfizetéssel mindössze bruttó 15 000 forintért meg lehet vásárolni, de ha elvisszük a Logiker Kft.-hez (➲ http://www.logiker.hu), ők mint garanciális Motorola-szerviz bruttó 2500 forintért frissítik a programját, amely így a GPRS-t is tudni fogja. A kételkedők kedvéért íme a történet: volt egyszer egy Motorola P7389-es sorozat, amely a megjelenése időpontjában egyedülálló tulajdonságokkal bírt: hangtárcsázás, IRDA, modem, WAP, diktafon stb. (körülbelül két évvel ezelőtt). Az ára is „szép” volt, ami természetesen az idő múlásával egyre lejjebb és lejjebb ment. A Vodafone nagyjából akkortájt rendelt belőle több ezernyit a Motorolától, amikor kifutott modellé vált. A Motorola-gyárban ekkor már nem

Néhány jó tanács a beállításokhoz

Hogyan állítsuk be egyszerűen a GPRS-t Debian GNU/Linux alatt? A létező legegyszerűbb megoldás, ha a `pppconfig` nevezetű programot választjuk, hiszen egy egyszerű `apt-get install pppconfig` parancs kiadásával telepíthető. Indítsuk el, és válasszuk a *Create a Connection* menüt. Mindent pontosan ugyanúgy csináljunk, mint amikor általános betárcsázást hozunk létre, a jellemzők azonban a következők legyenek:

```
User: vodawap
Passwd: vodawap0
Number: *99***1#
Method: PAP
ComPort: /dev/ircomm0
```

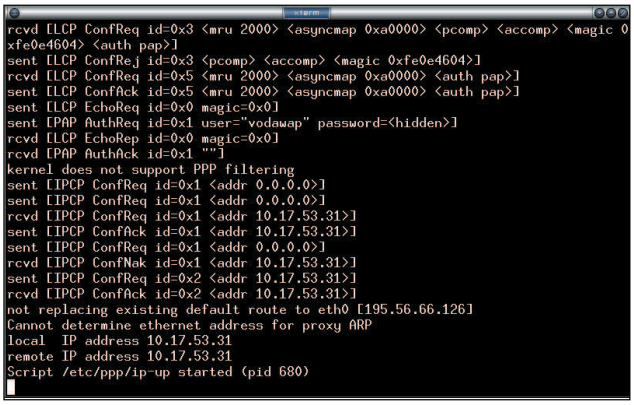
(vagy ahova a telefont kötöttük).
Ezek után *voda* néven mentsük el. Ilyenkor a program a */etc/chatscripts/* alá menti a modemmel kapcsolatos beállító alapparancsokat. A *voda* állomány közepén lesz egy sor:

```
"ATZ"
```

Ezt ki kell törölnünk, és helyette másik hármat kell elhelyeznünk:

```
"AT+CGDCONT=1,"P","vitamax.wap.vodafone.net"
" AT+CGQREQ=1,0,0,0,0,0;+CGQMIN=1,0,0,0,0,0"
" AT+CGATT=1"
```

Ezt követően a `pon voda` parancs kiadásával már is mehetünk a netre.



gyártották ezt a készüléket, de mivel nem akarták visszautasítani a megrendelést, a Motorola 260-as modell programját, amely tudta a GPRS-t, lebutították a P7389-es szintjére. Következésképpen, ha most befáradunk a fentebb említett Motorola-szervizbe, nagyjából 1–2 perc alatt működőképesé teszik a 260-as programot, így egy remek GPRS-es telefonunk lesz, összesen 17 500 forintért (a Vodafone-nál jelenleg bruttó tizenötezer forint a Vitamax City csomagban. Fontos dolog, hogy ez a trükk csak az új „Silver” készülékkel működik, a régebbiekkel sajnos nem.)
Érdemes a választásnál még arra is odafigyelni, hogy a telefon 3+1 csatornás modemmel legyen felszerelve, mivel így adatforgalmazás közben is fogadhatunk és kezdeményezhetünk hívásokat, természetesen csak akkor, ha van elegendő szabad csatorna. Ebben az esetben úgy tudunk rendszerfelügyeleti munkát végezni, hogy egyszerre vagyunk a hálózaton, és közben beszélgetünk. A fent említett Motorola ilyen telefon lesz a programfrissítés után.
Az IRDA-beállításáról nagyon jó leírást olvashatunk a

➲ http://www.pilot-link.org/howto/irhsync/ címen, ami bár angol nyelvű, de tele van képekkel.
A fenti tudnivalókat követően Dankó GPRS-HOGYAN-jának elolvasása után, ami a
➲ http://www.hup.hu//modules.php?name=News&file=article&sid=1311 címen érhető el, pár perc alatt a Világhálón lehetünk. Néhány jó tanács a beállításokhoz: a leírásban több helyen is megjelenő

```
OK 'AT+CGDCONT=5,"IP","internet.vodafone.net",
➥"",0,1'
```

parancs nem mindegyik telefon esetében működik, ugyanis jó néhány készülékben csak egy vagy két PDP context van, a leírásban pedig végig 5-re hivatkoznak. A Siemensnél és a Motorolánál itt egyszerűen át kell írni 1-re vagy 2-re, és máris működik.

Összegzés

Az átalánydíjas Vodafone GPRS-csomag csak azoknak ajánlható, akik nem tévesztik össze egy ADSL-kapcsolat sebességével, és akik számára nagyon fontos a mozgathatóság. A többi szolgáltatónál egyelőre nincsen átalány, de a rendszerfelügyeleti munka végzése során, utazás esetén még így is jobban megéri, mint az egyszerű betárcsázás.

Köszönjük a Vodafone Rt.-nek a tesztkártyát.

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

Kapcsolódó címek

Motorola-szakszerviz ➲ http://www.logiker.hu
GPR-beállítási HOGYAN ➲ http://www.hup.hu/modules.php?name=News&file=article&sid=1311
IRDA-HOGYAN ➲ http://www.pilot-link.org/howto/irhsync

Vigyünk haza atomórát a chrony segítségével! (1. rész)

Minden technojáték között, amely megmelengeti az igazi szakértők szívét, nincs még egy olyan nagyszerű, mint az atomóra.

Végre itt az eszköz, amelyről tudósok nemzedékei álmodtak: az ultrapontos időreferencia, a hihetetlen pontosságú időőrző eszköz. Nem egyszerű játékról van szó: néhány fizikai kísérlet, például a gravitációs elméletek ellenőrzése igen kis időtartamok mérését igényli.
A National Institute of Standards and Technology (NIST) Colorado államban egy boulderi laboratóriummal rendelkezik, ahol a hivatalos amerikai időt szolgáltató atomórákat működteti. Ebben a laboratóriumban a NIST-F1 Cézium-alapú atomóra szolgáltatja az időalapot 2×10^{-15} (két milliomod egymilliárdad része) céziumatom körülbelül 9 GHz-es rezgései alapján. Mindeközben már folyik a munka egy még ennél is jobb órán, ami egyetlen higanyion rezgéseit fogja mérni körülbelül az előző frekvencia 100 000-szeresén, ami ezerszeres pontosságnövekedést jelent majd.
Máris egyértelmű, mi a szent feladatuk az igazi pontosságmegszállott szakiknak. Csak annyit kell tenniük, hogy Linux-gépüket (vagy gépeiket) összehangolják egy ilyen őrülten hajszálpontos órával.
Természetesen nem mehetünk csak úgy be a számítógépboltba, hogy vegyünk egy atomórát (nem mintha nem próbáltam volna meg, és jellemző az eladó fickó pimaszságára, hogy egy rádium számlapos ébresztőórát próbált rámsózni). A második legjobb lehetőség a rádió-összehangolású óra, amiből számos modell áll a rendelkezésünkre. Ezeket a PC soros kapujához csatlakoztathatjuk, és az időjeleket a NIST-órával összhangban kapjuk.
De minek vegyünk új vasat, amikor jól megtervezett szabad programok ugyanúgy képesek erre a trükkre? Az NTP-t (Network Time Protocol) azért hozták létre, hogy a számítógépeket össze lehessen hangolni, illetve hogy időreferenciákat tudjunk hálózaton keresztül továbbítani. Az NTP-kiszolgáló olyan közel tartja az időt a referenciához, amennyire csak lehetséges. A távoli NTP-ügyfelek ezeket a kiszolgálókat kérdezhetik le, és hozzájuk igazíthatják a gép valós idejű óráját (real-time clock, azaz RTC). Az osztott rendszerek természete miatt az effajta időegyeztetés meglehetősen összetett feladat. A csomagok utazásának ideje két gép közt ugyanis valamilyen nem nulla változó értéket vesz fel. Az NTP-be különféle kiigazítási sémákat építettek be, hogy a változó lappangási időt számításba tudja venni.

Miért pont a chrony?

Nem csak egy NTP-ügyfél vagy -kiszolgáló létezik Linux alá. Az NTP használatának legegyszerűbb módja, ha magunk indítunk el egy programot, például az `xntpd`-t és egy NTP-kiszolgálóra irányítjuk. Csakhogy ez a program, illetve a legtöbb NTP-ügyfél akkor működik a legjobban, ha folyamatosan az Internethez csatlakozhat. Sajnos a legtöbben otthonról egyelőre csak időszakos modemes kapcsolattal tud az Internethez csatlakozni. Itt lép képbe a `chrony`. A `chrony` olyan program, amely kifejezetten az időszakos kapcsolatokat támogatja. Bár a telepítés és a beállítás érthető, elsőre kicsit riasztó lehet, ezért lépésenként megyünk végig a legáltalánosabb eseten: ha az otthoni felhasználó modemes kapcsolattal rendelkezik.
A `chrony` tartalmazza a ***chronyc*** nevű szöveges alapú ügyfélprogramot; a `chronyd` nevű, NTP-kiszolgálót, amely démonként futhat a háttérben; illetve néhány beállításfájlt. A `chronyd` démonnal (chronyd) a `chronyc` ügyfélprogramban kiadott parancsokkal tarthatjuk a kapcsolatot.

Letöltés és telepítés

Néhány Linux-terjesztés eleve tartalmazza a `chrony` valamely változatát. Ez a változat valószínűleg valamilyen régebbi darab, például 1.15-ös vagy még régebbi. Ebben az esetben az új változat telepítése előtt a régit töröljük.
Először is töltsük le a `chrony` tároló fájlt a honlapról (lásd a ***Kapcsolódó címeket***). Jelenleg legfrissebb száma a 1.16.1. Az 1.16 változatot az 1.16.1 folttal egészíthetjük ki. A forrást kicsomagoljuk, majd alkalmazzuk a foltot:

```
tar -zxvf chrony-1.16.tar.gz # A forrÆs
➥kicsomagolÆsa
cd chrony-1.16 # BelØp nk a chrony k nyvrtÆrba
gunzip < ../chrony-1.16-1.16.1-patch.gz |
➥patch -p1
patching file NEWS
patching file configure
patching file rtc_linux.c
patching file version.txt
```

A program `configure` parancsfájlt használ a beállítások gyors elvégzéséhez. Az egyetlen kapcsoló, amit nekünk kell kézzel beállítanunk, a telepítés könyvtára. Ezt a `--prefix` kapcsolóval tehetjük meg. Alapértelmezés szerint a `chrony` a `chronyc` ügyfelet a ***/usr/local/bin***, a `chronyd` démont pedig a ***/usr/local/sbin*** könyvtárba helyezi. Azaz a következő parancscsal egyenértékű:

```
# az elıbb is hasznÆlt chrony-1.16
➥k nyvtÆrban:
./configure --prefix /usr/local
```

Miután lefutattuk a `configure`-t, a `make` futtatása előtt megtisztíthatjuk a forrást. Hogy miért? Azért, mert a forrás tartalmaz néhány írásmódbeli ravaszságot, ami miatt a GCC előfeldolgozó panaszkodni fog. Ha most rögtön lefuttatjuk a `make` parancsot, igen sok ilyesféle üzenetet fogunk kapni:

```
warning: extra tokens at end of #endif
➥directive
```

Ez ugyan semmi rosszat nem jelent, de könnyen megoldhatjuk, hogy a program tisztán forduljon le. Szerkesszük át a ***regress.h***, ***reports.h*** és ***rtc_linux.h*** fájlokat. Az utolsó sorban találunk egy

`#endif`-meghatározást, amit egy konstans neve követ. Ezt a nevet kell megjegyzésbe tennünk. Például a ***report.h***-ban kell beírni a `sed "s:^#endif:#endif //:"*.h` sort, és a chrony máris gyönyörűen fordul.
Ezt követően a korábban használt chrony-1.16 könyvtárban adjuk ki az alábbi parancsot:

```
make
su root  # a telep tØshez rendszergazda kell
install
```

A soron következő lépés, hogy megvizsgáljuk, rendszerindításkor elindul-e a `chronyd`. Arra kell figyelni, hogy az új változatot pontosan a régi helyére tegyük. Egyébként több lehetőség között választhatunk. A legegyszerűbb módszer, ha a `chrony` leírásában található bekezdést a ***/etc/rc.d/rc.local*** fájlba illesztjük.

Stratum Conundrum

Itt az ideje, hogy beállítsuk `chrony`-t. Mivel gépünk óráját egy NTP-kiszolgálóval szeretnénk egyeztetni, ki kellene választanunk egyet. Tulajdonképpen több különböző kiszolgálót érdemes választani, hátha valamelyik elérhetetlen.
A lista különböző rétegekbe (stratum) szedve sorolja fel a kiszolgálókat. De mi is az a réteg? Itt most nem geológiai sziklarétegekről van szó, inkább a hagyma héjára kell gondolnunk. Az első réteg olyan kiszolgálókból áll, amelyek közvetlenül az atomórához igazodnak. A második réteg olyan NTP-kiszolgálókat jelent, amelyek az első rétegkiszolgálóktól kapják az időadatokat és így tovább. Az első réteget inkább hagyjuk békén, hacsak nem egy fizikalabort működtetünk, netán többezer gépet számláló saját hálózatunk van otthon. Az első réteg gépei fenn vannak tartva a másodlagos kiszolgálók időadat-szolgáltatására, illetve az olyan gépeknek, amelyeknek a második réteg által nyújtott néhány mikroszekundumos pontosság is elfogadhatatlanul kevés. A mi céljainknak egy második vagy akár harmadik rétegből választott kiszolgáló is tökéletesen megfelel.
Fontos, hogy az NTP-kiszolgálók rendszergazdáinak általában levélben kell jelezni, hogy az ő gépükhöz szeretnénk igazodni. Ne feledjük ezt megtenni. Képzeljük el, mi lenne, ha otthoni gépek ezrei kezdenének el időpontokat kérni egy szegény egyetemi NTP-kiszolgálótól! A rendszergazdának tudnia kell, hogy valóban órák összehangolásáról van-e szó, netán egy újfajta floodtámadás érte.
Eszményi esetben olyan NTP-kiszolgálót tudunk választani, ami nincs túl messze a gépünktől (IP-szempontból). Ez nem feltétlenül egyezik meg a földrajzi távolsággal. Ökölszabályként használhatjuk a `traceroute` kimenetét, amely felsorolja a csomópontokat, amelyeken a csomagok a gépünk és a cél között átmennek. Én például new yorki lakos lévén a következő gépeket választottam: ***ntp.ctr.columbia.edu*** a Columbia Egyetemről, ***clock.psu.edu*** a Pennsylvania Állami Egyetemről és ***ntp0.cornell.edu*** a Cornell Egyetemről (ha ezt a kiszolgálót használjuk, küldjünk egy levelet a ***pgp1@cornell.edu*** címre). Váltsunk át rendszergazdai módba, és a következő meghatározást adjuk a ***/etc/chrony.conf*** fájlhoz:

```
server ntp.ctr.columbia.edu offline
server clock.psu.edu offline
server ntp0.cornell.edu offline
```

Természetesen a kiszolgálónevek helyére azokat a NTP-kiszolgálókat kell írni, amelyek közel esnek a lakhelyünkhöz. Figyeljük meg, hogy a kiszolgálók meghatározása üzemen kívüli (off-line). A legtöbb modemkapcsolatú gép rögtön rendszerindításkor nem építi fel a kapcsolatot. Ezért amikor a `chronyd` elindul, nem szabad elkezdenie a kiszolgálók lekérdezését. Felhívjuk a figyelmet arra, hogy a `chrony`-leírás azt javasolja, hogy a nevek helyett az NTP-kiszolgálók számalakú TCP/IP-címeit használjuk, ezáltal csökkentve a DNS-től való függést. Nos, ennek ellenére a legtöbb NTP-kiszolgáló rendszergazdája mégis inkább DNS-alapú neveket akar használni, mivel így megtarthatják a kiszolgáló áthelyezésének képességét. Emellett modemes kapcsolatunkon remélhetőleg el tudjuk érni az ISP DNS-ét, így én azt javaslom, a ***chrony.conf*** fájlban mi is inkább az NTP-kiszolgálóneveket használjuk.

Jelszavak

Az NTP-protokoll támogatja a csomagszintű azonosítást (packet authentication). Végtére is ha egy céget vezetünk, nyilván nem szeretnénk, ha bárki bármilyen időt beállítgathatna a gépeinken. A hibás időponttal bekerülő pénzügyi bejegyzések megzavarhatják könyvelőinket (vagy könyvelőprogramjainkat).
A `chrony` egyszerű jelszóalapon oldja meg az azonosítást. Megadhatunk néhány számokkal azonosított `chrony`-felhasználót és jelszavakat rendelhetünk hozzájuk. Az ide vonatkozó ***chrony.conf*** fájlbejegyzés:

```
commandkey 9
keyfile /etc/chrony.keys
```

A fenti sorok azt mutatják, hogy ez a gép a 9-es szám alapján keres a ***/etc/chrony.keys*** fájlban tárolt jelszavak közt. Ez utóbbi például a következőt tartalmazhatja:

```
9 zack
```

Zack a macskám neve. A `chrony` használata előtt az egész házban ez a vad állat közelítette meg legjobban a pontos órákat. Minden reggel fél nyolckor – a hétvégéket is beleértve – szánalmasan nyávogni kezdett, amíg meg nem etettem. Hamar nagy gyakorlatra tett szert az ágyból való kiverésben.
Ezen felül a `chrony`-nak meg kell adnunk, hol tárolja, hogy mennyivel tér el vagy hogyan jár a gép órája az NTP-kiszolgálóidőhöz képest. Az eltérési fájl helyét a következő módon adhatjuk meg:

```
driftfile /etc/chrony.drift
```

Így a `chrony`-nak nem kell minden egyes indítás után összegyűjtenie az adatokat és újraszámolnia az eltérést.

A következő részben ezek óráinak összehangolásáról lesz szó.

Linux Journal 2002. szeptember, 101. szám

Fred Mora
Unix-rendszergazda és -fejlesztő az 1990-es évektől kezdve. Több könyv és műszaki kézikönyv szerzője és társszerzője. Fred az IBM-nél dolgozik.

chrony-honlap
➲ http://www.rrbcurnow.freeuk.com/chrony/releases.html

Bemutatkozik az AOLserver

Az AOLserver használata egyáltalán nem akkora kihívás, és nem is olyan bonyolult dolog, mint amilyennek először gondolnánk.

Az Apache, ez a jól ismert HTTP-kiszolgáló mindig is a nyílt forrású programok egyik reklámgyermeke volt: népszerű, megbízható, rugalmas, biztonságos, hordozható, bővíthető és megfelel az internetes szabványoknak. Az első megjelenése óta Apache-ot használok, és mindig élvezet volt dolgozni vele.

Nekünk, akik tudjuk, hogy számos nyílt forrású operációs rendszer, szövegszerkesztő, adatbázis és programozási nyelv létezik, talán az sem olyan nagy meglepetés, hogy nem az Apache az egyetlen nyílt forráskódú HTTP-kiszolgáló. Az viszont meglepő lehet, hogy az egyik ilyen választási lehetőség éppen az America Online-tól származik, attól a cégtől, amely a (szintén nyílt forrású) Mozilla böngészőt támogatja.

Az AOLserver sok tekintetben az Apache-hoz hasonló szolgáltatásokat nyújt: nyílt forrású felhasználási szerződés alatt fut, a beállítása könnyű és rugalmas, valamint a beágyazható (plugin) bővítmények írására API-felületet biztosít. Mivel azonban az AOLserver alapvetően más felépítésű, mint az Apache, bizonyos esetekben az előbbi hatékonyabb megoldás lehet. Továbbá az AOLserver beépített Tcl-értelmezővel, többszálú megoldással, adatbázis-API-felülettel és adatbáziskapcsolat-tárazással (database connection pooling) büszkélkedhet. Ha webhelyünk felhasználói nagyszámú adatbázis-kapcsolatot használnak, az Apache kiváltására esetleg az AOLservert is érdemes lehet szemügyre venni.

Ebben a hónapban az AOLservert fogjuk bemutatni – mintegy bevezetőként a nyílt forrású OpenACS (Open Architecture Community System) webalkalmazás-keretrendszerről szóló cikksorozatunkhoz. Bár az AOLserver nem feltétlen szükséges az OpenACS-hez, a rendszer beállításának és telepítésének ez az általánosan feltételezett módszere.

Keletkezéstörténete

Az AOLserver pályafutását NaviServer néven kezdte, amelyet a NaviSoft nevű, kiemelkedő minőségű, webterjesztéshez szánt ügyfél- és kiszolgálóoldali eszközeiről nevezetes, úttörő cég írt és árusított. Az AOL gyakran vásárolt meg olyan cégeket, amelyek érdekes módszereket fejlesztettek ki; a NaviSoftot az AOL inkább a kiszolgálóoldali fejlesztéseiért vette meg, semmint ügyféloldali eszközeiért.

Az AOLserver valószínűleg viszonylag ismeretlen maradt volna, ha nem következik be néhány fontos esemény: az AOL a program binárisait ellenszolgáltatás nélkül elérhetővé tette az Interneten, és ***Philip Greenspun*** dolgozni kezdett az ArsDigita Community Systemen. Az AOLserver segítségével könnyedén lehet nagy teljesítményű, adatbázisháttérrel támogatott honlapokat létrehozni; és az a tény, hogy az ACS igencsak keményen használ relációs adatbázisokat, azt sejttette, hogy az AOLserver tökéletes választás lesz.

Csakhogy míg maga az AOLserver ingyenes volt, a forráskódja továbbra sem volt elérhető a nagyközönség számára. 1999-ben Greenspun segített kidolgozni azt a megegyezést, amelynek értelmében az AOLserver 3.0 az AOLserver Public License alatt jelent meg, ami lényegében azonos a Mozilla Public License engedéllyel.

Ez év elején maga az ArsDigita lényegében kivonult az üzletből, legénységének maradványai pedig a Red Hat Software-hez igazoltak át. Az eredeti ACS program azonban továbbél az OpenACS projektben, amely az ACS eredeti Tcl-változata alapján készült – az AOLserver, illetve PostgreSQL vagy Oracle felhasználásával.

Az AOLserver nyílt forrású modellre való átvitele nem volt éppen zavartalan. Még a kezdet kezdetén néhány OpenACS-fejlesztő egy OpenNSD-nek nevezett saját ágat választott le (fork) az AOLserverből, arra hivatkozva, hogy az AOLserver-fejlesztőknek nyitottabbnak kell lenniük a közösség iránt. Az OpenNSD mostanra, úgy tűnik, elhalálozott, és az OpenACS-közösség ismét az AOLserver használatára buzdítja az embereket. Mindeközben az ArsDigita számos fejlesztést beépített az AOLserverbe, amelyek a kiszolgáló forráskódjában korábban nem szerepeltek, a legfrissebb üzembiztos vagy fejlesztői változatban sem. Minthogy számomra az AOLserver szinte mindig összekapcsolódott az OpenACS-sel, az AOLserver 3.3ad13 változatot fogom használni, amit a ➲ http://www.openacs.org/software.adp címen érhetünk el. A fejlettebb változatokat, ideértve a jövendőbeli 4.0-s fejlesztői változatát (snapshot) a ➲ http://www.aolserver.com címen érhetjük el. Jelenleg ezek a hivatalos változatok nem biztosítják az OpenACS támogatását.

Mitől is olyan nagyszerű?

Egészen ennek az évnek az elejéig, amikor a 2.0 végre megjelent a nagyközönség számára is, az Apache többfolyamatos kiszolgáló volt. Ez azt jelenti, hogy bármely időpillanatban számos Apache-folyamat futott, amelyek mindegyike az adott időben egyetlen HTTP-tranzakciót tudott csak lebonyolítani. Tíz egyidejű elérés kiszolgálása tehát tíz párhuzamos Apache-folyamat futását jelentette, 100 egyidejű kapcsolathoz természetesen 100 ilyen kapcsolatot kellett elérhetővé tenni.

Az Apache 2.0 ezen már némiképpen segít, mert lehetővé teszi, hogy egy folyamat több szálat futtass. Minden egyes szál egy-egy HTTP-kapcsolatot kezelhet, ami azt jelenti, hogy öt szálat használva az öt folyamatban összesen már legföljebb 25 egyidejű kapcsolatot tudunk fogadni. Minthogy a szálak általában kevesebb erőforrást fogyasztanak, mint a folyamatok, egy átlagos PC esetében ez tulajdonképpen teljesítménynövekedést jelent.

Az AOLserver ezzel szemben mindig is több szálon futott, és egyetlen folyamatot használt. Bármely időpillanatban mindig csak egyetlen `nsd`-példány fog futni a gépünkön, és a neve hűen tükrözi a tényt, miszerint valamikor NaviServer-démonnak hívták. Ez az egyetlen folyamat viszont számos egyidejű HTTP-kapcsolatot képes kezelni. Tulajdonképpen az AOL pontosan azért erőlteti tovább az AOLserver fejlesztését, mert az ennyire nagyszámú egyidejű kapcsolatot is képes kezelni.

Az AOLservert használják saját, nagy forgalmú webhelyeiken, ideértve a ➲ http://netscape.com-ot, az ➲ http://aol.com-ot és a ➲ http://digitalcity.com-ot.
A többszálú megoldás előnye – és lehetséges veszélye – az a tény, hogy a szálak könnyedén megoszthatják egymás közt az adatszerkezeteket. Az AOLserver ebből olyan módon kovácsol előnyt, hogy az adatbázis-kapcsolatokhoz létrehoz egy tárolót. Mivel az itt található kapcsolatok mindig nyitva állnak, webalkalmazásunknak nem kell időt fecsérelnie a megnyitásukra (vagy a lezárásukra). Továbbá minthogy csak ritkán fordul elő, hogy a kiszolgáló valamennyi HTTP-kapcsolata egy időben kérje az adatbázis-kapcsolatot, a tárolónak nem kell annyi kapcsolatot tartalmaznia, mint amennyi a szálak legnagyobb száma, ezáltal csökkenti a webkiszolgáló és az adatbázis-kiszolgáló által felhasznált memóriát. Elképzelhetjük úgy is, mint a csomagváltás (packet switching) adatbázisos megfelelőjét. A csomagváltás módszerének lényege, hogy a telefonvonalat számos fél között osztják meg egyszerre, kihasználva, hogy senkinek sincs szüksége a vonalra az idő száz százalékában.
Az AOLserver – akárcsak az Apache – támogatja a beágyazható modulokat (plugin modules). Számos modul létezik, kezdve az XML-értelmezőtől (`nsxml`) egészen a Python beágyazott változatáig (PyWx). Vannak modulok, amelyekkel CGI-programokat hajthatunk végre, vagy biztonságos kapcsolatot készíthetünk SSL-el, illetve az adott adatbázis-kezelőkkel, ilyen például a MySQL, PostgreSQL és az Oracle. Az OpenACS a PostgreSQL és az Oracle alatt is képes működni, és az ➲ http://openacs.org oldalról letölthető. Az AOLserver az `nsxml` modul mellett mindkét adatbázismodult tartalmazza.
Ahogyan a `mod_perl` lehetővé teszi, hogy a webfejlesztők az Apache beállításait és válaszait a C használata nélkül testreszabhassák, az AOLserver is nyújt egy beépített API-t, amelyen keresztül a testreszabható szolgáltatásokat Tcl nyelven állíthatjuk be. Az igazat megvallva én a fejlesztéshez szívesebben használnék Perlt vagy Pythont, de – mint azt számos AOLserver és OpenACS fejlesztő éveken keresztül bizonygatta nekem – a Tcl „nem olyan rossz”, így végül jó néhány érdekes, kezelhető alkalmazást készítettem a Tcl és az AOLserver segítségével, figyelemre méltóan rövid idő alatt (a beágyazott Python-modult még nem használtam, részben azért, mert az OpenACS megköveteli a Tcl használatát). Az AOLserver által nyújtott API meglehetősen megkönnyíti a munkát az olyasféle dolgokkal, mint a HTTP-fejlécek és a HTML-űrlapmezők értékei.

Fordítás és beállítás

Az AOLserver fordítása viszonylag egyszerű. Az Apache-csal ellentétben, amely a modulok fordítását a kiszolgáló telepítése után az `apxs` programon keresztül teszi lehetővé, az AOLserver esetében minden modult egyszerre kell lefordítani és telepíteni.
Bár külön felhasználó és csoport létrehozása nem kötelező, az AOLserver biztonsági okokból nem rendszergazdai jogosultsággal fog lefutni. Ezért az AOLserver fordítása és telepítése előtt nem árt, ha új felhasználót és csoportot készítünk a rendszeren – amennyiben követjük a hagyományt, akkor ***nsadmin*** néven. Az én Red Hat 7.2 rendszeremen egyszerűen ennyit kell tenni:

```
/usr/sbin/adduser nsadmin
```

Még mindig rendszergazdaként el kell készítenünk a ***/local/aolserver*** könyvtárat, ahová alapértelmezés szerint az AOLserver kerül. Ezt követően a könyvtár tulajdonjogát átadjuk az `nsadmin`-nak:

```
mkdir /usr/local/aolserver
chown nsadmin.nsadmin /usr/local/aolserver
```

Ha ez megvan, átváltunk `nsadmin` felhasználóra, megnyitjuk a ➲ http://openacs.org-ról letöltött forráskódot, és elkezdjük a fordítási folyamatot:

```
su - nsadmin
cd /tmp
tar -zxvf aolserver3.3ad13-oacs1-beta-
➥src.tar.gz
cd aolserver
./conf
```

A fenti sorok önműködően beállítják, lefordítják és telepítik az AOLservert a rendszerünk tulajdonságainak megfelelően, majd az elkészült fájlokat a ***/usr/local/aolserver*** könyvtár alá helyezik. Az AOLserver minden modult, amit csak tud, önműködően lefordít; figyelmen kívül hagyva (és kihagyva) az összes többit. Az én asztali gépemen, ahol a PostgreSQL fejlesztői könyvtárai megvannak, de az Oracle hiányzik, az AOLserver beállítása és telepítése után létrejön a PostgreSQL-meghajtó, de az Oracle-meghajtó egy az egyben hiányozni fog.
A fordítási folyamat némi időt vehet igénybe, és nem készít túl bőbeszédű kimenetet a képernyőn. Ha azt gyanítanánk, hogy a folyamat valahol lefagyott, belenézhetünk a ***log/aolserver.log*** fájlba; az összes fordítási kimenet itt található.
Amint befejeződött a létrehozási folyamat, a ***/usr/local/aolserver*** könyvtár alatt egy AOLserver másolatunk jön létre.
A ***/usr/local/aolserver*** alatti legfontosabb könyvtár a ***bin***, amelyben az AOLserver (`nsd`) program található, azokkal a megosztott könyvtárakkal (***.so***) egyetemben, amelyek a kiszolgáló egyes moduljaihoz fordultak le. A ***log*** könyvtár foglalja magában a kiszolgálóhoz tartozó elérési és hibanaplókat, végül a ***lib*** könyvtár tartalmazza a beépített Tcl-értelmezőt.
Az AOLserver beállításfájlja Tcl-ben íródott; alapértelmezés szerint egy egyszerű beállításfájl kerül a ***/usr/local/aolserver/sample-config.tcl*** könyvtár alá. Ezt a fájlt megvizsgálva látni fogjuk, hogy valamennyi beállítási direktíva tulajdonképpen Tcl változó-hozzárendelés. Ezek a változó-hozzárendelések szakaszokra tagolódnak, melyben minden szakasz általában egy-egy, a kiszolgálóhoz fordított modult jelent. Ahogy azt a példa-beállításfájlból is láthatjuk, a változókhoz közvetlen értékeket rendelhetünk. Például a következők szerint állíthatjuk át az AOLserver által figyelt HTTP-kaput 8000-re:

```
set httpport 8000
```

Mivel a beállításfájl Tcl-ben íródott, a `homedir` változót (amely alapesetben a ***/usr/local/aolserver***) az érték közvetlen kódolása helyett az AOLserverhez intézet kéréssel is beállíthatjuk:

```
set homedir        [file dirname [ns_info
config]]
```

Természetesen az egyszerű behelyettesítés segítségével változóbeállításainkat más változók értékére is alapozhatjuk:

```
set servername              "server1"
set pageroot
➥${homedir}/servers/${servername}/pages
```

Ez a két sor, amely az AOLserver példa-beállításfájljából

származik, a statikus URL-ek gyökerét a */usr/local/aolserver/servers/server1/pages* könyvtárra állítja. További beállításokat végezhetünk az `ns_param` paranccsal, amely általában két kapcsolót fogad: egy nevet és egy értéket. Minden kapcsolónak egy `ns_section` hívással kezdődő szakaszba kell tartoznia. Például a kiszolgáló hibakereső szolgáltatását úgy kapcsolhatjuk be, ha a teljes körű (global) a `ns/parameters` szakaszban bekapcsoljuk a `debug` kapcsolót:

```
ns_section "ns/parameters"
ns_param   debug            false
```

Sajnos az AOLserver kapcsolóinak leírása messze alulmúlja a hálózaton fellelhető Apache-dokumentáció szintjét. A kapcsolók csaknem teljes listáját az ➲ http://aolserver.com/docs/admin/config-reference.tcl.txt oldalon találhatjuk meg, ami egy olyan kiszolgálókiépítést mutat be, ami szinte mindent beállít. Ha befejeztük a rendszer beállítását, márpedig az alapértelmezett beállítások elég jók egy kisebb weblaphoz, az `nsd` program meghívásával és a használni kívánt beállításfájl megadásával elindíthatjuk az AOLservert:

```
cd /usr/local/aolserver
bin/nsd -f -t sample-config.tcl
```

A `-f` kapcsoló az előtérben futtatja az AOLservert, így a hibanapló a képernyőnkre kerül. Ha már elégedettek vagyunk azzal, ami történik, a `-f` kapcsolót eltávolíthatjuk – ettől kezdve kiszolgálónk hibanaplóját a *log* könyvtárban kereshetjük. Amennyiben azt szeretnénk, hogy az AOLserver a 80-as kapura hallgasson, mindenképpen rendszergazdaként kell indítanunk. Egyébként a Linux nem fogja elfogadni a kérelmet, azt üzeni nekünk, hogy kizárólag a szuperfelhasználó indíthat „kitüntetett” (azaz 1024-nél kisebb számú) kapukra hallgató kiszolgálókat. Igen ám, de ha csak a rendszergazda érheti el a 80-as kaput, az AOLserver viszont nem hajlandó rendszergazdaként futni, hogyan szolgáltathatunk mégis a 80-as kapun? Úgy, hogy az AOLservert rendszergazdaként indítjuk, és kapcsolóként megadjuk neki azt a felhasználót és csoportot, amelybe át kell váltania:

```
su root
cd /usr/local/aolserver
bin/nsd -f -u nsadmin -g nsadmin -t
➥sample-config.tcl
```

Ezek után böngészőnket átirányíthatjuk a *http://sajátgép.sajáttartomány.hu:8000/* címre, és már látjuk is a bemutatkozó AOLserver-dokumentumot, amely az új kiszolgálón üdvözöl bennünket. Figyeljük meg, hogy az AOLserver beállításfájljai egyaránt figyelembe veszik a számítógép nevét és az IP-számát, így ha csatlakozunk a hálózathoz, böngészőnket nem irányíthatjuk a *localhost* névre, hanem a teljes nevet kell használnunk.

Tcl-programok

Bár az AOLserver kétségtelenül kitűnő HTTP-kiszolgáló statikus dokumentumokhoz, igen valószínűtlen, hogy valóban csak erre szeretnénk használni. Sokkal gyakoribb eset, hogy a Tcl nyelv segítségével dinamikus lapokat készítenek. Ennek legkönnyebb és legegyszerűbb módszere egy HTML-kódot visszaadó Tcl-program készítése. A feladat végrehajtásához *.tcl* kiterjesztéssel hozzunk létre egy állományt a *pageroot* könyvtárban. Ide tetszés szerint bármilyen Tcl-t elhelyezhetünk; a legfontosabb dolog, hogy a végén adjuk ki az `ns_return` utasítást, azt az AOLserverben megadott Tcl-függvényt, amely három kapcsolót fogad:

1. a számszerű HTTP-válaszkódot (például 200 vagy 404), amely a program végrehajtásának sikerét vagy sikertelenségét hivatott jelezni;
2. a *Content-type* fejlécet, ami a visszaadandó tartalom típusát mutatja;
3. a felhasználóhoz visszatérítendő tulajdonképpeni tartalmat jelenti.

Például az egyszerű *Hello, világ* program így nézne ki:

```
ns_return 200 text/html "<html>
<head>
    <title>Testing</title>
</head>

<body>
     <p>Hello, world</p>
</body>
</html>"
```

Ha a fenti programot *hello.tcl* néven a *pageroot* könyvtárba helyezzük, majd betöltjük a böngészőbe, visszakapjuk az állomány betű szerinti tartalmát. Ennek az az oka, hogy nem állítottuk be újra az AOLservert, így a *.tcl* lapok használatát nem engedélyezi a *pageroot* könyvtárban. Az engedélyezéshez az `enabletclpages` kapcsolót a `ns/server/${servername}` szakaszban igazra (`true`) kell állítanunk:

```
ns_section "ns/server/${servername}"
ns_param   enabletclpages  true
```

Amint ezt a módosítást elvégeztük, az AOLservert újraindíthatjuk, és megpróbálhatjuk újra lekérni a *hello.tcl*-t. Ezúttal már a HTML-kimenetet kell látnunk a nyers text/plain kimenet helyett. A *.tcl* lap számos dolgot tartalmazhat: adatbázishoz kapcsolódhat, XML-állományokból szedhet ki adatot, hálózatokon keresztül szerezhet meg adatokat, illetve HTML-űrlapokból is kaphat adatot. Minthogy a Tcl számos karaktersorozat-kezelő parancsot ismer, igen sok lehetőségünk nyílik a bemenetet értelmezésére és a változók kimenetbe helyettesítésére.
Ne feledjük, hogy a Tcl az AOLserveren belül értelmeződik, nem egy külső folyamat hajtja végre. Ez egyben azt is jelenti, hogy az ilyen *.tcl*-fájlok sokkal gyorsabban hajtódnak végre, mintha az Apache-ban használnánk *.tcl* CGI-programokat; sok szempontból úgy viselkednek, mint a `mod_perl` Perl programjai.
Igaz ugyan, hogy a *.tcl*-fájlok nagyszerűek, ha a programozó HTML-kimenetet akar készíteni, csakhogy azt már keserű tapasztalataink alapján megtanulhattuk, hogy a grafikus tervezők általában nincsenek felkészülve arra, hogy forrásfájlokban megbúvó HTML-kódot javítgassanak. Ezért aztán sok webfejlesztő áttért a sablonok használatára – legyen az ASP, JSP, HTML::Mason, DTML vagy valamilyen hasonló módszer.
Az AOLserver saját beépített sablonrendszerrel rendelkezik, melynek neve ADP (AOLserver Dynamic Pages), és a felépítése gyanúsan hasonlít a Microsoft ASP nyelvére. A végrehajtandó kód <% és %> jelek közé kerül, a HTML közé ékelődött értékeket visszaadó kódot pedig <%= és %> jelek fogják közre. Például:

```
<% set foobar "abcdef" %>

<head>
     <title>Testing</title>
</head>
<body>
     <p>Hello, world</p>
     <p>Hello, <%= $foobar %></p>
</body>
</html>
```

A néhány lapnál nagyobb webhelyek a Tcl-kód egy részét valószínűleg megszeretnék osztani. Ennek a legkönnyebb módja, ha egy egy vagy több függvényt meghatározó ***.tcl***-fájlt készítünk, amely(ek)et az AOLserver induláskor betölthet. Ezek a függvények aztán valamennyi ***.tcl***- és ***.adp***-lapról elérhetők lesznek. A szolgáltatás engedélyezéséhez a következőket kell ***sample-config.tcl*** fájlunkhoz adni:

```
ns_section ns/server/${servername}/tcl
ns_param   library \
  ${homedir}/servers/${servername}/tcl
ns_param     autoclose    on
ns_param     debug  true
```

mivel a kiszolgálónk neve ***server1***, minden ***.tcl***-fájl, amit a ***/usr/local/aolserver/servers/server1/tcl*** alá helyezünk, be fog töltődni, amint az AOLserver elindul. Én például a következő tartalmú ***foo.tcl*** fájlt helyeztem ebbe a könyvtárba:

```
proc return_hello {} {
     return "hello"
}
```

Ezután újraindítottam az AOLservert (ami elengedhetetlenül szükséges ahhoz, hogy a Tcl-könyvtárfájlokat beolvashassa), majd a ***hello.adp***-t a következők szerint módosítottam:

```
<% set foobar "abcdef" %>
<% set hello [return_hello] %>

<head>
     <title>Testing</title>
</head>

<body>
     <p>Hello, world</p>
     <p>Hello, <%= $foobar %></p>
     <p>Hello, <%= $hello %></p>
</body>
</html>
```

Ettől kezdve a `hello` változó a `return_hello` függvényem eredményét tartalmazza, ami jelen esetben nem más, mint a hello szócska.
Mivel a könyvtár Tcl-függvényeinek betöltéséhez újra kell indítani az AOLservert, az új függvényt többnyire egyszerűbbnek találtam ADP-lapjaim `<% %>` szakaszában megadni. Csak miután már láttam, hogy függvény jól működik, tettem át a meghatározását a Tcl könyvtárba, és csak egyszer indítottam újra az AOLservert.
Az ADP és ***.tcl***-lapok minden dinamikus tartalmú oldalhoz igen jól használhatók. Néha azonban előfordul, hogy a lemezen dokumentum létrehozása nélkül szeretnénk valamilyen programot URL-hez rendelni. Könnyen megoldhatjuk a dolgot: egyszerűen csak egy Tcl-függvényt kell a ***tcl*** könyvtárfájlba helyezni, majd ezt a függvényt a kiválasztott URL-hez az AOLserver `ns_register_proc` parancsával bejegyezni:

```
proc http_hello {} {
  ns_return 200 text/html "<html>
                    <head><title>Hello!
                      </title></head>
                    <body><p>Hello!
                      </p></body>
                    </html>"
}

ns_register_proc GET /hello http_hello
```

Ha a fenti Tcl-kódot a korábban megadott programkönyvtár-könyvtár valamelyik fájljába helyezzük, majd újraindítjuk a kiszolgálót, és a böngészővel a ***/hello*** címre keresünk, a `http_hello` függvényünk kimenetét fogjuk látni.
Már jó néhány éve programozok `mod_perl`-ben, mégis le vagyok nyűgözve attól a könnyedségtől, ahogyan az AOLserver és az `ns_register_proc` függvény segítségével új lapokat illeszthetünk be. Ráadásul a `GET` és `POST` kérelmekhez különböző függvényeket rendelhetünk. Akár azt is megtehetjük, hogy olyan szűrőfüggvényeket jegyzünk be, amelyek egy másik lap által készített kimenetet figyelnek vagy változtatnak meg.

Összefoglalás

Ha egy hálózati feladatban OpenACS-t szeretnénk használni, szinte biztosan meg kell tanulnunk az AOLserverrel dolgozni. Még ha az OpenACS nem is érdekel valakit, az AOLserver rugalmassága, gyorsasága és többszálú képessége megéri, hogy egy kis figyelmet szenteljünk neki, hátha felhasználhatjuk dinamikus lapjainkhoz.

Linux Journal 2002. szeptember, 101. szám

Reuven M. Lerner (reuven@lerner.co.il) egy kisebb webes és internetes módszerekkel foglalkozó tanácsadó cég tulajdonosa és vezetője. Az ATF honlapon érhető el (➲ http://www.lerner.co.il/atf/).

Kapcsolódó címek

Az AOLserver-honlap, amelyet a SourceForge tart fenn a ➲ http://www.aolserver.com címen érhető el.

Az AOLserver API Python-változatát a ➲ http://pywx.idyll.org címen találjuk.

Az OpenACS-közösség fóruma ➲ http://openacs.org/bboard

Végül *Philip Greenspun* 1999-es AOLserver témájú cikkei kitűnően bemutatják a módszert. A négyrészes sorozat első részét a ➲ http://www.linuxworld.com/linuxworld/lw-1999-09/lw-09-aolserver_1.html címen olvashatjuk.

Kicsi, de jól működő Linux-doboz

Nincs mindig a legnagyobb, a leggyorsabb gépre szükségünk ahhoz, hogy jól végezzük el a ránk bízott feladatot.

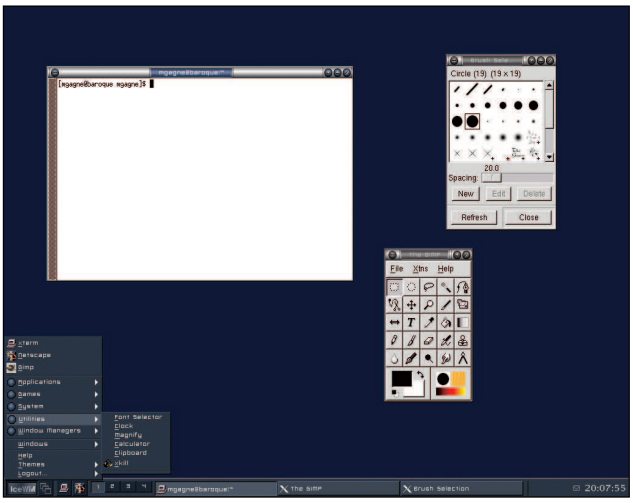
Jóízű ételek csak jó konyhában születhetnek, François, és nagyon egyszerű, hogy mi tesz naggyá egy konyhát. Az átlagon felüli konyha a tehetség és a körülmények megfelelő összjátékából születhet meg, ezt a két dolgot pedig a szakács és eszközei testesítik meg. Rendkívüli dolgok alkotásához ugyanúgy nincs szükség a világ legnagyobb konyhájára, mint ahogyan a Linuxszal való főzéshez sincs szükség a leggyorsabb gépre. Nézzük, mon ami, ezt az IceWM nevű ablakkezelőt! Bár egy meglehetősen szerény méretű programcsomagról van szó, mégis vonzó megjelenésű, könnyen és rugalmasan munkára fogható.

François, te nem is figyelsz arra, amit mondok! Quoi? Á, isten hozott, mes amis, Chez Marcel kitűnő Linux-konyhájában. Amíg François kihozza a bort, megmutatom az asztalotokat. Vite, François! Szaladj le a borospince északi szárnyába, és hozd fel azt az 1999-es Napa Valley Cabernet Sauvignont, amit ma már egyszer minőségi ellenőrzésnek vetettünk alá. Amíg a bor érkezésére várunk, hadd fájdítsam a szíveteket a mai menü kapcsán néhány gondolattal. Az ember könnyen hagyja magát elragadtatni egy (vagy akár több) szupergyors processzorral, hatalmas tárterülettel és szinte végtelen memóriával felszerelt bíró gép ábrándja által. Legtöbbünket sajnos nem ilyen géppel áldotta meg a sors. Akik esetleg emlékeznek még pénztelen egyetemista napjaikra, bizonyára jól tudják, hogy gyakran azt használtuk, amihez épp hozzájutottunk. Nézzük, hogyan használhatjuk ki egy egyszerűbb géppel a Linux képességeit!

Előételként egy olyan Linux-rendszercsomagot vizsgálhatnánk meg, ami a grafikus felületet teljesen mellőzve egyetlen hajlékonylemezen elfér, de ezt a megoldást szeretném elkerülni. Elképzelésem egy vonzó megjelenésű munkaasztal néhány barátságos grafikus eszközzel, amelyek még a korlátozott erőforrások megszorításai között is megfelelően működnek.

Következzen néhány érdekes számadat: új Red Hat 7.3-as munkaállomásomon a KDE 3.0-s munkaasztalt egyetlen `xterm`-mel futtatva a `free` parancs hozzávetőlegesen 34 000 KB felhasznált memóriát jelez (nem számítva az átmeneti és gyorstár méretét). Ezt az értéket az alaprendszer által használt memóriát levonva kaptam, és nem futtattam a KDM-hez hasonló belépéskezelő programot sem. Az igaz, hogy a KDE sok új szolgáltatása fut, így az elhalványuló eszköztippek, előugró ikonok, hangtémák és a többi, csakhogy ez az alapértelmezett beállítás. A Gnome 1.4 szintén egyetlen `xterm`-mel a 27 000 KB-ot közelíti. Ezek a Gnome- és KDE-értékek igen éles ellentétben állnak az IceWM futtatásakor kapott mindössze 7500 kilobájtos értékkel.

Mint láthatjátok, már a kezdetekkor óriási különbségek adódhatnak, amikor grafikus környezet futtatásába fogunk. A ***Marco Macek*** által írt, majd ***Mathias Hasselmann*** által továbbfejlesztett IceWM egy kisméretű, pehelysúlyú, mégis gazdag szolgáltatáskészlettel rendelkező ablakkezelő. Támogatja a több munkafelületes asztalt, egér nélkül is kezelhető,



1. kép Az IceWM egy nagyon dögös ablakkezelő

lehetővé teszi asztali témák használatát (tartalmaz is néhányat), és meglehetősen jó munkát végez annak a bizonyos másik operációs rendszernek az utánzásában is.

Az IceWM legfrissebb változatát a 40. CD Magazin/Fogado könyvtárában találjátok meg. A weboldal előre fordított RPM-csomagokat is biztosít, de a program fordítása is igen egyszerűen, a hagyományos ötlépéses módszerrel végrehajtható:

```
tar -xzvf icewm-1.0.9-2.tar.gz
cd icewm-1.0.9-2
./configure
make
su -c make install
```

Bizonyára testre akarjátok majd szabni saját IceWM-környezeteteket, és hozzáadjátok azokat az egyéni vonásokat, amelyek otthonossá teszik a rendszert. Ennek legjobb módja a helyi ***.icewm*** könyvtár létrehozása (a saját könyvtáratokban) és a beállítófájloknak ebbe a könyvtárba történő átmásolása. Az alapértelmezett telepítés ezeket az állományokat a ***/usr/X11R6/lib/X11/icewm*** könyvtárba helyezi:

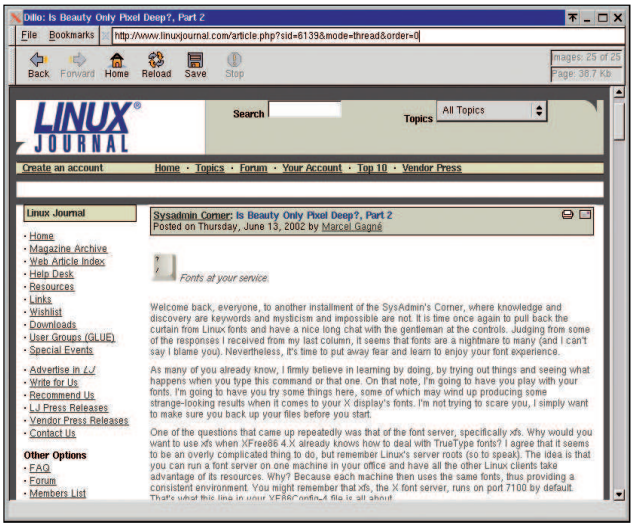
```
mkdir $HOME/.icewm
cp -r /usr/X11R6/lib/X11/icewm/* $HOME/.icewm
```

Et voilà! Készen is állunk új ablakkezelőnk futtatására. Az indítás legegyszerűbb módja egy ***.xinitrc*** fájl létrehozása a saját könyvtáratokban. Ennek egyetlen sort kell tartalmaznia, amely így fest:

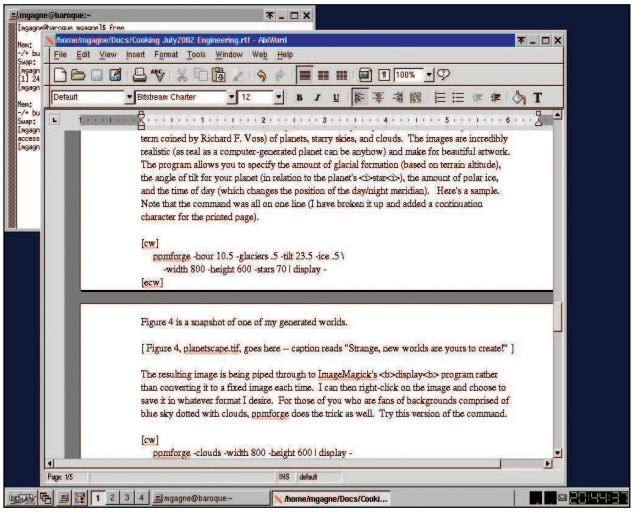
```
exec icewm
```

Most gépeljük be a `startx` parancsot, és máris korcsolyáz-

hatunk az Ice ablakkezelő jegén. Kattintsatok a bal alsó sarokban lévő menüre! Néhány program elindítása után asztalotoknak az *1. képen* láthatóhoz hasonlóan kell kinéznie.
Néhány alkalmi matematikai művelet elvégzése után az én IceWM-példányom (csak az `xterm`-et futtatva) megelégszik megközelítőleg 7500 kilobájt felhasználásával. Nem is rossz. Most nézzük, milyen területen lehetne hasonló megtakarításokat elérni. Mi a helyzet például a böngészőprogramokkal? Úgy tűnik mostanában, hogy a Világhálót böngésző programok egyre nagyobbak és nagyobbak lesznek. Az rendben van, hogy emellett egyre sokoldalúbbak is, de ha nem egy szupergyors számítógépen dolgozunk, talán hajlandóak lennénk lemondani néhány szolgáltatásról. Még az Opera is, ami egyébként kitűnő böngésző és az utóbbi időben sokat javult (részben a sebesség és a méret terén), egy kicsit túlsúlyosnak bizonyulhat.
Érdemes megfontolni ***Jorge Arellano*** Dillo nevű böngészőjének a használatát, amely jó választási lehetőséget nyújt a mai „mindent egyben" stílusú nagyobb és csillogóbb böngészőkkel szemben. Nemcsak azt vehetitek majd észre, hogy a rendszerrel szemben igen szerény követelményeket támaszt, de örvendetes az oldalakat megjelenítő sebessége is. Vessetek egy pillantást a *2. képre*, amelyen egy igen eleven, működés közbeni Dillót látható.



2. kép A Dillo: kicsi és gyors grafikus webböngésző



3. kép Az AbiWord munka közben

A Dillo használatának megkezdéséhez először a program honlapját kell a ➲ http://dillo.cipsga.org.br címen felkeresni, majd a legfrissebb forráskódot letölteni. A kicsomagolás és fordítás a már ismerős lépésekkel zajlik:

```
tar -xzvf dillo-0.6.6.tar.gz
cd dillo-0.6.6
./configure
make
su -c make install
```

A fordítás és telepítés után a program a `dillo &` parancs begépelésével indítható. A Dillo elindítása után újra ellenőriztem gépem erőforrásait és megállapítottam, hogy a program futásának teljes folyamata mindössze 848 kilobájt RAM-ot igényel, ami meglepően kis számérték.
Kétségtelen, hogy az iroda legfontosabb asztali alkalmazása a szövegszerkesztő. Milyen választásunk lehet a minden szükséges tudással bíró, az oly elterjedt MS Word formátumot is ismerő szövegszerkesztők terén? Ezeknek a tulajdonságoknak egy kis helyre való összepréselése lehetetlen kérés, egyetértek, de hogy hangzik az AbiWord 1700 kilobájtja? Hasonlítsuk össze az OpenOffice Writerrel, amelynek indításakor a rendszer közel 12 000 KB-ot eszik. Amennyiben csak egy szövegszerkesztőre van szükséged és géped erőforrásai végesek, látogass el a ➲ http://www.abisource.com oldalra, ahonnan e nagyszerű program egy ingyenes másolatát töltheted le.
E címen számos előre fordított állomány érhető el, valószínűleg nem sok tennivalód akad majd az üzembe helyezéssel. Akik inkább forráskódból szeretnek dolgozni, néhány további lépésre lesz szükségük, de még így is nagyon egyszerű az egész:

```
tar -xzvf abiword-1.0.2.tar.gz
cd abiword-1.0.2/abi
./autogen.sh
./configure
make
su -c make install
```

A telepítés befejeztével a program az `abiword &` paranccsal indítható. Egy figyelmeztetés: az AbiWord indításakor foglalkoznunk kell még egy gyakori kifogással a betűtípusokat illetően. Alapértelmezésben az AbiWord betűkészletei nem szerepelnek az X betűkészlet-kiszolgáló listájában elérhető betűtípusok között. Ezt az AbiWord az indításakor szóvá is teszi. A program így is futni fog, csakhogy a vele kapott betűtípusok nem lesznek elérhetők. A hiba helyrehozásához a betűkészletek elérési útvonalát egyszerűen hozzá kell adni az X betűkészlet-kiszolgáló útvonallistájához (`font path`). A Red Hat (vagy a hozzá hasonló) rendszeren a legegyszerűbb módszer a `chkfontpath` parancs használata:

```
chkfontpath -a /usr/local/share/AbiSuite/fonts
service xfs restart
```

A fenti elérési útvonal azt feltételezi, hogy az AbiWordöt forráskódból telepítettük. Más rendszereken a */etc/X11/fs/config* állomány szerkesztésére lehet szükség, ekkor az elérési útvo-

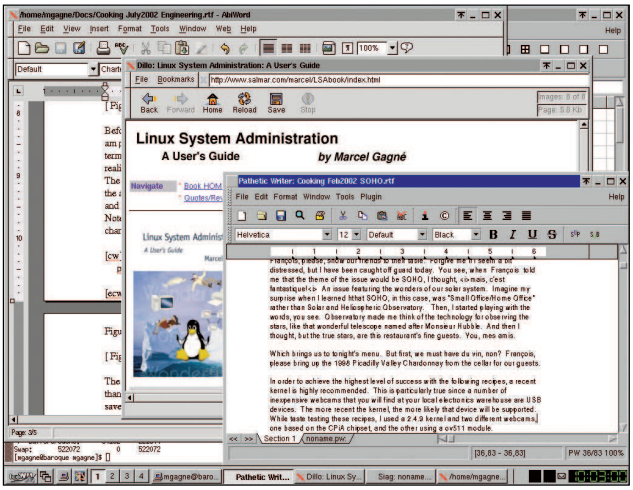
nalat kézzel kell hozzáadnunk. Keressük meg azt a bekezdést, ami így kezdődik:

```
catalogue = \
/usr/X11R6/lib/X11/fonts/75dpi:unscaled,
...
```

Figyeljetek arra, hogy az utolsót kivéve minden sor vesszővel végződik. Ha a betűtípusok elérési útvonalát kézzel adod a listához, a második sortól kezdődően ne felejts el vesszőt tenni a sorok végére, és az utolsóról eltávolítani. Ezután indítsd újra az `xfs` szolgáltatást, majd az ablakkezelő munkafolyamatát.
Így már van egy pici, de ügyes ablakkezelőnk, egy apró, de gyors böngészőprogramunk, valamint egy szerény, de hatékony szövegszerkesztőnk. Mit szólnátok egy teljes irodai programcsomaghoz? Tekintetetekből azt olvasom ki, hogy azt hiszitek, Chef Marcel túl sokat kóstolgatta a saját borát.
Vessetek hát egy pillantást a Siag Office nevű csomagra, amelynek neve a „Scheme in a grid” (Minta a hálóban) rövidítése. Valóban nem túl találó név egy irodai csomag számára, de ez az a program, amit kár lenne kihagynotok. Ez a jól összegyúrt csomag tartalmaz többek között egy szövegszerkesztőt (amelyet szerzője csak Pathetic Writernek – szánalmas írónak – nevez), egy táblázatkezelőt (Scheme in a grid) és egy Egon nevű animációkészítő programot. Rögtön az elején el kell mondanom, hogy a program az MS Word formátumát közvetlenül nem támogatja. Ez sokakat elrettenthet, mások számára viszont nem okoz gondot, hiszen az rtf formátum alkalmas a dokumentumcserére. Szükségetek lesz a ***XawM*** programozói könyvtárakra (ez egy Athena-megfelelő könyvtár), a ***Mowitz*** könyvtárakra (more widgets – vagyis még több bigyó), és végül magának a Siag Office csomagnak a letöltésére és fordítására. Bízzatok bennem: egyszerűbb a dolog, mint amilyennek látszik, a csomagok a honlapról elérhetők. Mindegyik üzembe helyezhető a kicsomagolás-lefordítás hagyományos ötlépéses eljárásával. Például a XawM könyvtárral a következő lépésekre van szükség:

```
tar -xzvf XawM-1.5u.tar.gz
cd XawM-1.5u
./configure
make
su -c make install
```

A három csomag telepítése után a `pw` paranccsal el kell tudnotok indítani a szövegszerkesztőt, a `siag` paranccsal pedig a táblázatkezelőt.
Korábban vizsgáltuk az egyes programok memóriaigényét, nézzük, milyen igényt támaszt a rendszerrel szemben a Pathetic Writer! Pereghetnek a dobok: a Pathetic Writert futtatva és egy egyhasábos cikket betöltve mindössze 1300 kilobájtot használtam fel!
És tényleg: az IceWM, az AbiWord, a Pathetic Writer, a Siag táblázatkezelő, a Dillo böngésző egyidejű futtatása egy `xterm`-mel a memóriából még mindig csak kevesebb mint 32 MB-ot igényel. Vessetek egy pillantást a *4. képre*, amelyen ezekkel a pehelysúlyú programokkal telezsúfolt képernyőkép látható.
Természetesen ha erőforrásaink korlátozottak, az alkalmazások csak a teljes kép egy részét alkotják, hiszen bármelyik rendszercsomag alapértelmezett telepítése nagy valószínűséggel egy sor felesleges szolgáltatást fog futtatni. Fel kell tennünk



4. kép Mindez kis helyen is elfér

magunknak a kérdést, hogy szükségünk van-e olyan programok futására, mint a Sendmail, az NFS, az Apache kiszolgáló és a többi, ha számítógépünket munkaállomásként használjuk. Egy `ps axfw` parancsot futtassatok, és döntsétek el, hogy szükségetek van-e ezekre a szolgáltatásokra.
Mint láthatjátok, nincs szükség a legújabb, legnagyobb vagy leggyorsabb gépre ahhoz, hogy tapasztalatot szerezzetek a Linux-rendszerről. Egyszerűen csak vennetek kell a fáradságot, hogy a pillanatnyilag legnépszerűbb csomagok mögé tekintsetek és feltegyétek magatoknak a kérdést: valóban szükségem van erre a regimentnyi szolgáltatásra? Gyakran a kisebb, a szerényebb bizonyul használhatóbbnak. Mégis, ha a borokról van szó, egy szép testes pohárnyi sosem árt. Habár, az a nagy pohár hatással lehet a teljesítményetekre (a taxik az étterem előtt fognak várakozni). Ürítsük poharainkat, és élvezzük ezt a nedűt!
A következő havi viszontlátásra! A votre santé! Bon appétit!

A cikkben szereplő programok megtalálhatóak a 40. CD Magazin/Fogado könyvtárában.

Linux Journal 2002. szeptember, 101. szám

Marcel Gagné
Mississaguában, Ontario államban él. Ő a szerzője a Kiskapu kiadásában szeptemberben megjelent Linux-rendszerfelügyelet (ISBN 963-9301-40) című könyvnek (jelenleg is egy könyvön dolgozik). A mggagne@salmar.com címen érhető el.

Kapcsolódó címek

AbiWord-honlap ➲ http://www.abisource.com
Dillo webböngésző ➲ http://dillo.cipsga.org.br
IceWM (Ice Window Manager) ➲ http://www.icewm.org
Marcel „borlapja”
➲ http://www.marcelgagne.com/nfwine.html
Siag Office ➲ http://siag.nu

Álomgépek

Vajon mit fog gondolni a következő nemzedék a mi jó öreg kiszolgálóinkról és asztali kövületeinkről?

Néhány éve, mielőtt még a Web ennyire népszerű lett, az időszerű álomgép mindig valamilyen kiszolgáló volt. Ezekbe a gépekbe örökké a legújabb, leggyorsabb processzorok kerültek (akár több is), egészen elképesztő mennyiségű memóriát tartalmaztak (nagyjából 64 MB-ot), hatalmas és szupergyors SCSI-merevlemezekkel működtek (két-három lemezzel), hogy elosszák a terhelést, és soha nem általános (Intel) rendszerek voltak, hanem DEC, SUN Sparc, IBM RS vagy HP-UNIX kiszolgálók. A régi szép időkben az adatközpontokat is a Wang és hasonló cégek szállították. Manapság ha kiszolgálót telepítek, egy kisebb gép tökéletesen megteszi; 1 GHz-es processzor, 128 MB memória, 18 GB-os IDE-merevlemez kerül a házba, ami a régi MFM- vagy RLL-meghajtókhoz képest szinte repül – és még ez is felesleges. Mostanában az álomgépet a főnök asztalán kell keresni. Természetesen neki nincs ideje arra, hogy tíz másodpercet várjon az Outlook vagy a Netscape elindulására, és a legfrissebb híreket is okvetlenül meg kell néznie a CNN-en, miközben hatalmasra hízott szövegszerkesztőjével dolgozik – amelynek szolgáltatásai az átlagos felhasználó számára nagyjából 95 százalékban ismeretlenek. A mai grafikus kártyákon több memória van, mint tárhely az első merevlemezemen. A számítógépek pedig még mindig csak gyerekkorukat élik. Csak néhány év kell, és visszatekintve értetlenkedve fogjuk rázni a fejünket: hogy is tudtunk ilyen lassú ócskavasakkal dolgozni?

Find++

Az emberek már csak ilyenek: az egyszerű, könnyen használható dolgokat kedvelik. Az sem árt, ha ezek a dolgok grafikus felületen futnak. A Find++ segítségével szavakat vagy mondatokat kereshetsz a merevlemezeden, akár az állományok nevében, akár magukban az állományokban. Találat esetén – ha a fájltípushoz program is hozzá van rendelve – a megfelelő alkalmazást elindítva megnyithatod a fájlt. Ennél egyszerűbb már nem is lehet. Futtatásához szükséges: libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm és glibc.
➲ http://ios.free.fr/?page=projet&quoi=15

dnstracer

Adott egy tartománynév – vajon honnan származik? A `dnstracer` segítségével az összes hivatalos adatot egészen a forrásig visszakövetheted. A program rengeteg lekérdezést szabályozó beállítással rendelkezik. Futtatásához glibc szükséges.
➲ http://www.mavetju.org/unix/general.php

CheckInstall

Szinte hihetetlen, hogy még soha nem írtam erről a programról, miközben nap mint nap használom (a cikkeimben szereplő programokat kivétel nélkül magam fordítom). Az alkalmazást a `make install` helyett használom, ha valamilyen csomagot forrásból telepítek. Elég egyszerű módszerrel ugyan, de RPM-, DEB- és TGZ- (Slackware) csomagokat készít. Így könnyedén figyelemmel kísérhetjük a gépen található programokat, miközben lázasan csomagokat telepítünk és távolítunk el. Még a teljesen szűzen telepített gépeimen is használom, amelyekre az RPM és a `checkinstall` is hamar felkerül. Futtatásához bash és glibc szükséges.
➲ http://asic-linux.com.mx/~izto/checkinstall

LGeneral

Sok-sok évvel ezelőtt néha összejöttünk a haverokkal, és például a NATO Division Commander nevű játékkal játszottunk. Jó ideje nem játszottam ilyesmit, ez a játék azonban felelevenítette az emlékeimet. Sajnos most már nincs időm arra, hogy egy teljes hétvégén át nyugodtan ücsörögve stratégiai játékokat játsszak, de még ha volna is, alighanem másra fordítanám. Az LGames viszont bármikor, akár álmatlan éjszakákon is elővehető, csak ne felejtsük el levenni a hangerőt. Futtatásához szükséges: libSDL_mixer, libSDL, libpthread, glibc, libm, libdl, libvorbisfile, libvorbis, libogg, libsmpeg, libartsc, libX11 és libXext.
➲ http://lgames.sourceforge.net

CRM

A CRM segítségével különféle eseményeket, feladatokat kísérhetünk figyelemmel, kioszthatjuk őket munkatársainknak, határidőket szabhatunk meg, és figyelhetjük a megoldás folyamatát. Ha szolgáltató céget tartasz fenn, érdemes kipróbálnod. A tervezetekhez riasztást is beállíthatsz, ha nem akarsz kicsúszni a határidőből. Könnyű telepíteni, könnyű használni. Futtatásához szükséges: MySQL, Apache PHP- és MySQL-támogatással, webböngésző.
➲ http://www.it-combine.com/crm

ippl (Internet Protocol Logger)

E hónapban nem volt könnyű a három évvel ezelőtti kínálatból választani, ugyanis még mindig elérhető több kiváló alkalmazás, ám közülük talán az `ippl` a leghasznosabb. Ha figyelemmel kell kísérned hálózati forgalmad jellegét, az `ippl` lehet a megfelelő eszköz. Szépen fejlődött az elmúlt évek alatt. Talán a legjobb dolog benne az, hogy beállítható kizárólag azoknak a protokolloknak a naplózására, amelyek forgalmát meg akarod figyelni. Hátránya, hogy a megszokott TCP-, UDP- és ICMP-protokollon kívül mást nem támogat, de még így is jól jöhet. Futtatásához libthread és glibc szükséges.
➲ http://pltplp.net/ippl

A cikkben szereplő programok a 40. CD Magazin/Falatka könyvtárában megtalálhatóak.

Linux Journal 2002. szeptember, 101. 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.