Gyakran mondják, hogy azért ne használjunk magyarításokat, mert az angol jövevényszavakkal sokkal pontosabban ki tudjuk fejezni magunkat egy-egy új vagy gyorsan fejlődő területen. Sokszor találkozunk viszont olyan szavakkal, amelyek még az angolban is pontatlanok vagy több hasonló értelemmel bírnak. Ezen szavak áthozása gyakran nagyobb kavarodást okoz, mintha mi magunk vezetünk volna be helyettük újakat. Lássunk néhány ilyen területet!

Ebben a hónapban több cikkben is előkerül a munkafelületek kérdése. Egyszerűbb esetben nincsen gond, hiszen az ember egy adott gép előtt ül, a monitoron látszik valami, és csak ezt az egy dolgot kell névvel ellátni – hívhatjuk mondjuk: asztalnak. A gond akkor kezdődik, amikor igyekszünk elmagyarázni, hogy a valóság egy kicsit bonyolultabb, hiszen a gép elindul, majd az egyik monitorkártyán megjelenít egy grafikus felületet. Maga a felület nem „tud” semmit, csupán azt adja, amit a neve ígér: egy grafikus elemek megjelenítésére képes felület. Ezek után a felületen programok jelennek meg, vagy – többnyire – először egy ablakkezelő indul el, amely meghatározza, hogy a programok milyen ablakban fussanak. A helyzetet tovább nehezíti, hogy az ablakkezelők között is vannak lényegesen bonyolultabb rendszerek, mint amilyen például a KDE és a Gnome; hogy ezeket megkülönböztessük, gyakran nem ablakkezelőnek, hanem munkafelületnek hívjuk őket.

Szerencsére itt azért könnyen rendet teremthetünk, nem úgy, mint a site szó esetén. A szó eredetileg egyszerűen „hely” értelemmel bír. Ezt a szót nagy örömmel használják angol nyelvterületen a website rövidítéseként, így gyakran találkozunk a site-webhely párosítással. Igen ám, de ugyanígy használják a site szót az egy vállalaton belüli telephelyekre is, az pedig furán néz ki, ha a cég harmadik kerületi webhelyéről beszélünk.

Vagy ott van például a „virtuális”. A szónak ma már elég nagy „hagyománya” van nyelvünkben, valami olyat jelent, ami nem valós. Ezzel párhuzamban a Virtual LAN jelentése egy olyan hálózat, amely nem „létezik” önmagában. Erre a fogalomra igyekeztünk jó megfelelőt találni, amikor a VLAN-okról szóló cikkben sokszor használtuk a képzett hálózat kifejezést. Elsőre talán ugyanúgy furán hangzik, mint ahogy a socket fordítására a foglalat hangzott pár évvel ezelőtt.

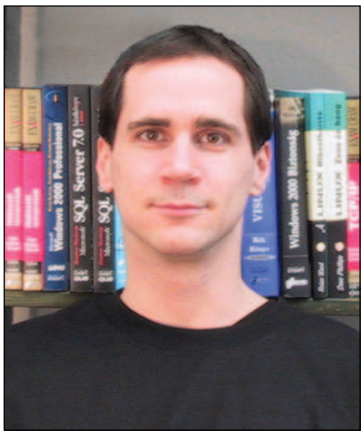
Sokszor van, hogy egy angol kifejezésre több magyar fordítás is születik, sőt versenyeznek egymással, amíg valamelyik „kihullik”, mint ahogy (véleményem szerint szerencsére) kihullott például az ottlap. Az egyik ilyen szónál a mi fejünk is fő, történt ugyanis, hogy a regular expression fordításaként a szabályos kifejezést és a szabványos kifejezést is használtuk. Kérdés, hogyan lehetne ötletesen rövidíteni egy-egy ilyen magyarítást, valahogy úgy, ahogy az angol eredetiből regex lett. Egyik munkatársunk az „szkif” rövidítést javasolta. Mi a véleményetek róla?

De ezek a viták tovább is gyűrűzhetnek. A minap hallottam egy fura elméletet, miszerint is az útválasztó és a router nem ugyanaz, mivel a router sokkal okosabb. Ezért ne is használjuk az útválasztót, hanem a rútert (a helyesírási szabályok szerint). Emlékszem, régen sokat vitatkoztunk, hogy a routerre miért van két fordítás (akkor még a forgalomirányító is „versenyben volt”) – most hirtelen az angol router szó új értelmet nyert? Talán egyszer maguk a szakemberek is eljutnak oda, hogy megegyezzenek abban, mi micsoda.

---

alagút – tunnel.
burkoló – wrapper.
chat forum → csevegőfórum
csevegőfórum – chat forum.
csíkozás – striping, a RAID-eszközöknél a RAID-0-megoldás, több eszközre tárol egy fájlrendszert, melyek között csíkonként osztja szét az adatot, így a műveleteknél a két eszköz egyszerre dolgozik. Nincs hibatűrés, de megfelelő eszközök használatakor jelentős sebességnövekedést eredményezhet.
desktop environment → munkakörnyezet.
ham – a levélszűrő rendszerekben használják a spam párjaként, hasonló hangzása miatt. Minden levél „ham”, ami nem minősül kéretlen levélnek.
hely – site.
hop → ugrás.
jelelosztó – (hálózati elosztó) hub, network hub.
kapcsoló – (hálózati kapcsoló) switch, network switch.
kéretlen levél – spam.
képzett LAN – virtual LAN, olyan hálózat, ami egy másik hálózat „tetején” létezik, általában valamilyen titkosítást használó rendszer, ami a létező hálózatot használva, attól független hálózatot képez.
kötet – volume, a fájlrendszereknél a lemezrészeket gyakran „kötetekként” használják, így lehetőség van a lemezrész átméretezésére, bővítésére stb. Lásd még a 45. oldalon található „RAID és LVM alapfokon“ című cikket.
levélszemét – spam.
mirroring → tükrözés.
munkakörnyezet – desktop environment, a ~ jellemzője, hogy nemcsak ablakkezelő, hanem további kiegészítésekkel, programokkal rendelkezik.
owner → tulajdonos.
property – tulajdonság.
regex → regular expression.
regular expression → szabályos kifejezés.
router → útválasztó.
sablon – template.
section → szakasz.
segmentation → szakaszolás.
site → hely, néha webhely vagy telephely rövidítéseként is használják.
spam → levélszemét, kéretlen levél.
striping → csíkozás.
switch → kapcsoló.
szabályos kifejezés – regular expression, azaz regex. A Unix világában nagyon sok helyen használhatunk ilyen kifejezéseket, amelyekkel valamilyen mintára illeszkedő szövegrészekre hivatkozhatunk.
szakasz – section, segment.
szakaszolás – segmentation.
szkif – szabályos kifejezés.
template → sablon.
tükrözés – mirroring, a RAID-eszközöknél a RAID-1-megoldás, az adatokat „tükrözve”, többszörösen tároljuk, így az egyik tárolóeszköz meghibásodása esetén sem veszítünk adatot.
tulajdonos – owner.
tulajdonság – property.
tunnel → alagút.
ugrás – hop, a hálózatoknál a csomagok több eszközön, gépen keresztül haladnak („ugrálnak”) végig. Fontos tulajdonsága lehet például egy csomagnak, hogy hány ugrásból jut el a célállomáshoz.
útválasztó – router.
Virtual LAN → képzett LAN.
volume → kötet.
wrapper → burkoló.

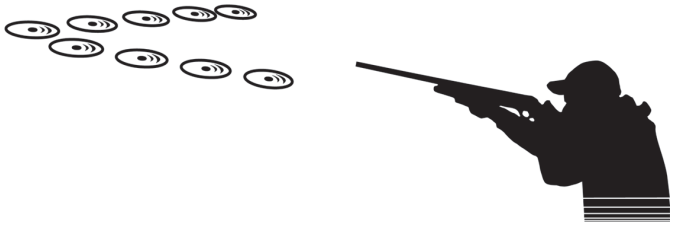
Beköszöntő



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

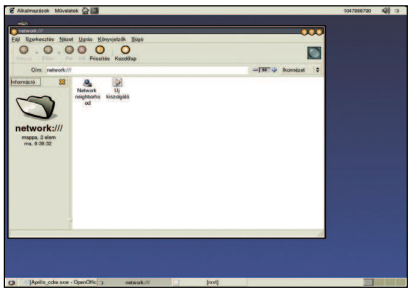
Ahogy az emberek felélednek téli álmukból, majd egy keveset leadnak a télen felszedett kilókból húshagyó keddtől fogva locsolkodásig, valamint nem utolsó sorban az egyetemisták átverekedték magukat a vizsgaidőszakon, egyre több idő jutott az egyéb feladatokra. Nálunk is hasonló tapasztalható, e havi számunkban ugyanis a szokásosnál is több cikk követelt magának helyet. Néhány sajnos kimaradt, viszont cserébe olyan témákkal is foglalkozunk, amelyeket hosszabb ideje ígérünk.
Ilyen például több munkaállomás felélesztése egyetlen gépen. Hosszú ideje keresek ötletes és jó megoldást erre a feladatra, aminek nagyobb irodákban vehetjük igazán jó hasznát. Mindig is tiszteletet éreztem azon rendszergazdák iránt, akik egymaguk igyekeztek tizenöt–húsz gépet karbantartani, heroikus küzdelemmel telepítgetve az újabb csomagokat és programokat minden egyes gépre külön-külön. Különösen nehéz azoknak a feladata, akik tizenöt-húsz különböző alaplappal, processzorral és egyéb elemekkel ellátott gépet igyekszenek kordában tartani.
Természetesen nem mondhatjuk el, hogy az irodákban egyik napról a másikra átvette a hatalmat a GNU/Linux, mint ahogy nem is várhatjuk az emberektől, hogy egy-két nap alatt átszokjanak egyik munkafelületről a másikra. Érdekes viszont, hogy a webmesterek között mindig is nagy táborra talált a Linux, hiszen számukra minden eszközt biztosít. A nekik szóló Kovácsműhely e hónapban négy cikket is tartalmaz, és külön öröm számomra, hogy három cikk magyar szerzőtől származik.
De ugyanígy egy magyar szerző, ***Kolcza Péter*** írása a 45. oldalon található cikk, amely a komolyabb kiszolgálóknál szinte kötelező RAID-rendszerek alapjait igyekszik ismertetni. De gyakorló rendszergazdák számára talán sokkal izgalmasabb a hosszú évek óta komoly háborúvá fejlődött küzdelem fegyvertárának egyik ígéretes eszköze, a levélszemetek szűrésére alkalmas Spambayes csomagot bemutató cikk. Nem mondom, hogy kis rendszereknél rövid távon nagy haszonnal tudjuk alkalmazni, de egy komolyabb levélforgalmat bonyolító kiszolgálónál már érdemes elgondolkodni a használatán.
Sajnos ma már komoly küzdelmek folynak ezen a területen. Hiába mondják a „szemetelők", hogy „ez marketing, és a házak postaládájában is mennyi kéretlen reklámot találunk" – szerintem az Interneten keresztül küldött számolatlan levélszemét teljesen más csoportba sorolható. Az elején az ember még elfogadja, hogy szegény cégnek nincs pénze reklámozni, leveleket küld, az szinte ingyen van, de amikor már napi öt levelet kapok, amelyben szexéhes hölgyikék csak arra várnak, hogy kielégítsem őket, és még nyolcat, amelyben az ehhez szükséges pénisznagyobbítás olcsó és gyors módszerét javasolják, kiegészítve néhány Viagra-ajánlattal, akkor bizony feszültté válok. Ezért is zavar, hogy egyre több magyar cég él ezzel a reklámlehetőséggel.
Remélem, hogy kedvenc rendszerünk hamarosan hatékonyan kiszűri majd ezeket a leveleket, és talán nincs messze az az idő sem, amikor a mesterséges értelem az irodákban talán éppen ezen a területen mutatkozik be először: egy programozott titkárnő, aki nagy örömmel dobálja ki az „Ön nyert" leveleket, és diplomatikusan, de határozottan ismételgeti a telefonba: „Igen, értem, hogy ez egy rendkívüli ajánlat, de idén sajnos nem tervezünk számítógépparkunk számára ötvenrészes törlőkendőt beszerezni."
Addig is, kitartást kívánok mindenkinek!

Programvadászat



Gnome 2.2 Woodyhoz

A KDE mellett a legnépszerűbb ablakkezelő rendszer a Gnome. Mindkét rendszernek elkötelezett hívei vannak, ezzel is megosztva a linuxos társadalmat. Mostani korongunkra a Gnome 2.2 került fel a Debian/GNU Linux megbízható változatához, a Woodyhoz. Használatához azonban mindenképpen frissítenünk kell az XFree86-ot is, az ehhez szükséges csomagok szintén megtalálhatók a CD-n.



Ezek úgynevezett „backport” csomagok – mivel a Woodyban nem állnak rendelkezésünkre a megfelelően friss csomagok, valaki a következő (Sarge) vagy a mindig fejlesztés alatt álló csomagjait ülteti át több-kevesebb sikerrel a megbízható változatra. Jelen esetben is ez történt, a lelkes debianosok nekiálltak, és elkészítették a Gnome-csomagokat és az XFree86 frissítést, hogy az üzembiztos változatok kedvelői is élvezhessék azokat az előnyöket, amiket esetleg csak a következő rendszerváltásnál kapnának meg.

Telepítés

A csomagfrissítés menete kissé zilálttá teheti rendszerünket, úgyhogy erre mindenképpen készüljünk fel némi nyugtató segítségével. Valószínűleg a legprimitívebb módon kezdtem neki a frissítés-telepítés eljárásnak: rendszergazdaként kiadtam a `# dpkg -i /cdrom/GNOME2.2/XFree86*.deb` parancsot, erre a telepítő nekiugrott a rengeteg fájlnak, és elkezdte a csomagokat frissíteni, illetve telepíteni. Ezt a módszert nevezhetnénk legbutábbnak is, ami szintén igaz (ezzel két „leg”-et ütöttem egy csapásra).
Természetesen jobban járunk, ha az `apt` által használt adatbázishoz hozzáadjuk a csomagokat, erre az `apt-cdrom -a add` parancsot használjuk, majd az `apt-get install gnome` paranccsal telepítjük a kívánt csomagokat.

A használat

Akik szeretik a kattintgatós ablakkezelőket, azok biztosan ezt is kedvelni fogják, de a grafikai elemek miatti memóriafelhasználás természetesen jóval felülmúlja az általam igen kedvelt AfterStep ablakkezelőjének igényét, irodai célokra használva azonban a KDE versenytársa lehet. Nagyon jó a nyelvi váltás kezelése, bejelentkezéskor kiválaszthatjuk a kívánt nyelvet, onnantól kezdve a programok nagytöbbsége a kiválasztott nyelven szól hozzánk. Ami külön tetszett, az a hálózat kezelése: az ***Alkalmazások-Hálózati kiszolgálók*** menüpontról indulva elénk tárul a „Network Neighborhood”, azaz a hálózaton lévő többi számítógép.
Nem tetszett, hogy az ***.rtf*** fájlhoz nem tudtam hozzárendelni az OpenOffice.org programot, akármivel próbálkoztam is. A másik gyenge rész a testreszabás volt.
Összességében, aki szereti az 1.4-es Gnome-ot, az ezt is szeretni fogja!
Az Internetről történő telepítéshez az alábbi sorokat adjuk hozzá az ***sources.list*** fájlhoz:

```
deb http://mirrors.evilgeniuses.
➥org.uk/debian/backports/
➥woody gnome2.2/
deb-src http://mirrors.
➥evilgeniuses.org.uk/debian/
➥backports/woody gnome2.2/
```

A 4.2.1-es XFree86-hoz:

```
deb http://people.debian.org/
➥~blade/woody/i386 ./
```

XFree86 4.3.0

A Linux grafikus felületének alapját jelentő XFree86 újabb változata jelent meg, amit forrás és előre fordított csomagokban is megtalálhatunk a CD-mellékleten. Lássuk, mi is változott!
A grafikuskártya-meghajtók:

- ATI Radeon 9x00 2D-támogatás, 3D-támogatás a Radeon 8500, 9000, 9100, és M9 lapkakészlethez.
- Támogatás az i810-hoz 845G, 852GM, 855GM és 865G integrált lapkákhoz, beleértve a 2D, 3D (DRI) és XVideo-támogatást is.
- Az nVIDIA nForce2 integráltlapka-meghajtó, illetve GeForce 4, GeForce FX-támogatás.
- SiS-meghajtók frissítése a legújabb lapkákhoz.
- 2D-gyorsítás a Trident CyberBladeXP/Ai1 lapkákhoz.
- Az OpenGL 1.3 és DRI meghajtótámogatás 4.0.4-esre frissült.

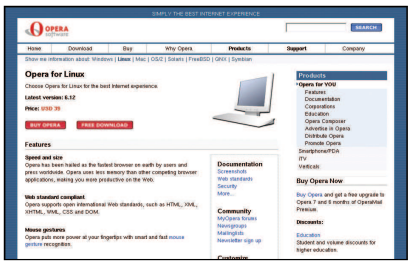
I18N és betűk:

- A FreeType2-frissítés 2.1.1-re változott, a TrueType (OpenType/TTF), OpenType/CFF -és Type 1-betűtámogatás is bekerült.
- Egy `mkfontscale` kiegészítés is csináltak, ezt a ***fonts.scale*** fájlok készítésére használhatjuk.

Természetesen a fenti lista nem teljes, akit érdekel az összes változás, az keresse fel az alábbi weboldalt:
➲ http://www.xfree86.org/4.3.0/RELNOTES.html

Biztonsági hibákat javít a legújabb linuxos Opera-változat

Korongunkon az Opera 6.12-es böngésző linuxos változata található. Az új Opera 6.12-es nem tartalmaz újdonságokat, csupán az előző változatokban



felfedezett hibákat javították ki. Ezek között található:

- az átirányítási sebezhetőség,
- az Xft könyvtáron lévő hiba,
- a betűkészletek kezelése és az élsimításban rejlő hibák javítása.

Az Opera könyvtárban megtalálható a ***.deb***, ***.rpm*** és ***.tgz*** csomagokban.
➲ http://www.opera.com

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

Kékfogúak

A Mitsumi két új USB-Bluetooth csatolót jelentett be, pontosabban egy csatolót két változatban. Az apróság segítségével az USB-kapun keresztül bármely számítógép könnyedén csatlakoztatható egy Bluetooth alapú hálózatba. A Class 1-es változat 100, a Class 2-es pedig tíz méter hatótávolságú, ennek megfelelően az előbbi drágább is, negyven dolláros társával szemben 64 dollárt kell kiadni érte. Mindkét csatoló 723 kb/s sebességre képes, telepítésüket és más eszközökkel való hálózati kapcsolatuk létrehozását a mellékelt Widcomm program segíti.
➲ http://www.mitsumidirect.com

NeuRosetta

A Digital Innovations Neuros nevű zenelejátszója egyelőre csak annyiban nyílt, hogy linuxos számítógépekkel is könnyedén összekapcsolható – igaz, ez is fontos előrelépés, hiszen más termékekkel ellentétben a Linux támogatása gyári lehetőség, nem valamilyen utólagos tákolás eredménye. Ennél sokkal nagyobb dobás lesz azonban a Neuros frissített változata, ami ugyancsak „hivatalosan" fogja támogatni az Ogg Vorbis formátumot. A Vorbis kodek ugyan egyes készülékeken már használható, de ez lesz az első eset, amikor maga a gyártó is támogatja a nyílt hangformátumot. A közös tervezetet a Xiph.org – a kodek fejlesztését irányító háttéralapítvány – és a Digital Innovations NeuRosetta névre keresztelte; illetve érdemes a Positron nevet is megjegyezni, mert ez lesz az a program, amelyik a linuxos számítógép és a Neuros közötti összehangolást végzi el.
A Neuros lejátszó kétféle változatban lesz elérhető, az egyik típusba 128 MB kapacitású memóriakártya, a másikba 20 GB-os merevlemez kerül. A készülék képes lesz rá, hogy az éppen lejátszott zenét URH tartományban szétsugározza, amely így akár hagyományos, például autós rádióval is fogható lesz. Beépített rádióvevőjével venni, illetve rögzíteni is tudja majd a rádióadásokat. Az Ogg Vorbis formátum használatát egy várhatóan májusban megjelenő, külön letölthető gépszintű programfrissítés teszi majd lehetővé.
➲ http://www.neurosaudio.com

Szemeiteket Ázsiára vessétek

Közel száz japán, koreai és kínai mérnök szövetkezett egy ázsiai Linux-változat készítésére. A szakemberek elsősorban az üzembiztosságot és a biztonságot szem előtt tartva szeretnének olcsó, a Microsoft termékeivel a kiszolgálók piacán versenyképes operációs rendszert létrehozni.
Mindeközben a Microsoft sem tétlenkedik: a NATO, Oroszország és Nagy-Britannia után Kína kormányzatával is tető alá hozott egy Government Security Program szerződést. A szerződés értelmében a programhoz csatlakozó szervezet vagy állam részleges hozzáférést nyer a Windows forráskódjához és a vele kapcsolatos műszaki adatokhoz, így meggyőződhet a rendszer biztonságosságáról.
A GSP program a Microsoft megbízható számítástechnikai kezdeményezésének része. Ami a jelek szerint nemcsak a megbízhatóságról, de arról is szól, hogy a nyílt rendszerek hibái a forráskód elérhetőségének köszönhetően viszonylag könnyen felderíthetők, és a saját kódjait zártan őrizve a cég – ebből a szempontból – egyértelműen hátrányba kerülne, amikor például egy kormányügynökségnél a felhasználandó operációs rendszer vagy program kiválasztására kerül sor.
Ázsiában a Microsoft termékei jelenleg meghatározó részesedéssel rendelkeznek mind a kiszolgálók, mint az asztali számítógépek piacán.

Ésszerű lépés

Az IBM 2,1 milliárd dollárért megvásárolta a Rational Software Corporationt. A programfejlesztők számára különféle modellező és tesztelő eszközöket, szolgáltatásokat kínáló, hazánkban is ismert cég felvásárlását az IBM még tavaly jelentette be, ám a művelet – a részvényesek jóváhagyása és a hatóságok engedélyének megszerzése után – csak nemrég fejeződött be. Az IBM elnökhelyettese, ***Steve Mills*** szerint a Rational minden (rendszerfejlesztési) részletre kiterjedő termékkínálata remekül ki fogja egészíteni a négy nagy IBM-márkanevet (WebSphere, DB2, Lotus, Tivoli). A Rational 89 országban volt jelen, alkalmazottainak száma meghaladta a 3400-at, eszközeit 600 ezer fejlesztő alkalmazza. Az IBM nem véletlenül vetett szemet a cégre: az IDC előjelzései szerint az alkalmazásfejlesztői programok piaca 2006-ra a jelenlegi kilencről 15 milliárdosra nő.
➲ http://www.rational.com

ELC Platform Specification

Az Embedded Linux Consortium (ELC) elkészült az ELC Platform Specification első változatával. A szabálygyűjtemény a beágyazott operációs rendszerekhez nyújt egységes alapot, így a fejlesztők időt és költségeket takaríthatnak meg, amikor termékeiket újabb szolgáltatásokkal ruházzák fel, illetve teljesen új készülék fejlesztésébe vágnak bele.
Beágyazott készülékeket az élet rengeteg területén, sokféle célra alkalmaznak. Ez is oka annak, hogy beágyazott számítógépeken számos különféle operációs rendszert futtatnak. Ezek egy része saját fejlesztésű, teljesen zárt, míg például a Linuxot választó cégek nyílt forrású alapokon építkezhetnek. Éppen ez a sokszínűség akadályozza ugyanakkor a fejlesztőket a saját kódjaik újrafelhasználásában vagy más készülékbe való átültetésében.
Az ELP a különféle Linux-terjesztések API-ját próbálja egységesíteni, olyan széles körben elfogadott szabványokra építkezve, mint az LSB 1.2, IEEE POSIX és a Single UNIX Specification. Így a fejlesztők kevésbé kötődnek majd egy-egy terjesztéshez, munkájuk felgyorsul, újdonságaikat hamarabb piacra dobhatják.
Az ELC tagjai között olyan nagy cégeket találunk, mint az IBM, a Matsushita, a MontaVista, a Red Hat, a Samsung vagy a Sharp.
➲ http://www.embedded-linux.org

Összejöttek, összedugták

A DSL Forum bejelentése szerint sikeres volt a több gyártó részvételével megtartott ADSL2 „plugfest", vagyis a megjelentek sikeresen összedugták azt, amit magukkal hoztak… Az ADSL2 sikeréhez – a többi szabványhoz hasonlóan – elengedhetetlen a különböző gyártók eszközeinek maradéktalan együttműködése. Hasonló próbát legközelebb nyár elején tartanak, illetve hamarosan az ADSL2+ eszközök kipróbálására is sor kerül.
Az ADSL2, ami az ITU ajánlásai között is szerepel, a jelenleg használt ADSL-szabvány továbbfejlesztése. Kiterjedt vonaldiagnosztikai eszközöket tartalmaz, képes az energiatakarékos működésre és a beállítások üzem közben történő módosítására. Az ADSL2-kapcsolatok sebessége – természetesen a központtól mért távolságtól függően – akár a 20 Mb/s-ot is elérheti.
➲ http://www.dslforum.org
➲ http://www.dsllife.com

Kiszolgálható VMware 4.0

Linux és Windows alá egyaránt letölthető a VMware 4.0 ingyenes próbaváltozata. Az új változatban számos újdonság található, a program magjától kezdve a hálózatkezelésen keresztül egészen a felhasználói felületig jó néhány dolgot megváltoztattak. A 4.0 támogatja a Red Hat Linux 8.0 és 8.1 próba, a Red Hat Linux Advances Server 2.1, a SuSE Linux 8.1 és Enterprise Server 8, a Mandrake Linux 9.0 és a Microsoft Server 2003 (próbaváltozat) operációs rendszereket. Bővültek a vendég-operációsrendszeren belül elérhető hibakeresési lehetőségek, a vendégrendszerből is lehet CD-lemezt írni; újdonság – többek közt – a VESA BIOS-, a DirectDraw- és az USB 2.0-támogatás. Hasznos szolgáltatás, hogy a vendég-operációsrendszer állapotáról bármikor pillanatfelvétel készíthető, és ebbe az állapotba később visszaállítható.

➲ http://www.vmware.com

Új ATI tévékártyák

Az ATI bejelentette, hogy Európában is megkezdi kedvelt All-In-Wonder Pro 9700 típusú kártyáinak a forgalmazását. Az Észak-Amerikában is közkedvelt kártya a cég legnagyobb teljesítményű,

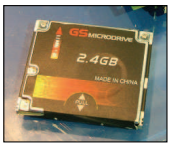
DirectX 9.0-s támogatást nyújtó Radeon 9700 Pro lapkájára épül. Alkalmas tévéadások vételére és 720×576 képpont felbontással, MPEG-2 formátumban történő rögzítésére, rádiós távirányítóval rendelkezik, és természetesen filmek szerkesztésére alkalmas programcsomag is jár hozzá.

Az ATI ugyancsak most vezeti be az európai piacra All-In-Wonder VE PCI termékét. A 64 MB DDR RAM-mal szerelt kártyákat elsősorban az olcsó, sokszor AGP-foglalat nélkül készülő számítógépekbe szánja a gyártó.

➲ http://www.ati.com

Magicstor

A kínai GS Microdrive Inc. Compact Flash Type II felületre illeszkedő, a Hitachi Microdrive meghajtójához nagyon hasonló kisméretű merevlemezt mutatott be. Mivel a Microdrive név védelmet élvez, az apró lemezt Magicstor névre keresztelték. Eleinte csak OEM-vásárlók számára fogják értékesíteni a távol-keleti régióban, ám később önmagában is megvásárolható lesz, elsőként Japánban és az Egyesült Államokban. Bolti ára 230–250 dollár körül alakul majd, ami az 1 GB-os Microdrive 100 ezer forint körüli árát tekintve kedvezőnek mondható; OEM-vásárlók pedig 150 dollár körüli áron is megszerezhetik. A 2,4 GB kapacitású egységből év végére 1,5 milliós havi termelést szeretnének elérni, illetve év végéig egy 3,6 és egy 4,7 GB méretű változat is megjelenik. A cég nemcsak fényképezőgépekben és MP3-lejátszókban szeretné viszontlátni termékeit, hanem megkezdte egy kifejezetten mobiltelefonokba szánt változat fejlesztését is.



Beelőz a Windows Server 2003

Némely helyeken – jóleső, kéjes gonoszságtól sem mentesen – azon csámcsognak, hogy a Windows Server 2003 megjelenésével a Microsoft a Datacenter, az Enterprise és a Standard mellett egy Web Edition nevű változatot is piacra dob. Az újfajta, viszonylag olcsón, 399 dolláros áron megvásárolható változat legfeljebb 2 GB memóriát és két processzort kezel majd, ügyfélfelhasználási szerződések telepítését nem igényli. A Microsoft tájékoztatása szerint nem lehet majd az összes csatornán keresztül beszerezni, de hogy ez pontosan mit jelent, azt egyelőre nem tudni.

A Netcraft felmérései szerint a Microsoft megoldásai évek óta 20–25 százalékos részesedéssel bírnak a webkiszolgálók piacán, amit sokan túlságosan gyenge eredménynek tartanak. Természetesen a maga hatvan százalék körüli részesedésével messze az Apache vezet, ám csak nézőpont kérdése, hogy a pohár félig teli vagy félig üres-e. A 20–25 százalék ugyanis bőven elegendő ahhoz, hogy a Microsoft megszerezze az előkelő második helyezést a rangsorban. Új termékével pedig egyáltalán nem esélytelen arra, hogy a Unix világától tartózkodó, elsősorban kisebb webhelyeket üzemeltető ügyfelei között számottevő sikert érjen el.

➲ http://www.netcraft.com/survey

Strapabíró TabletPC

A különlegesen ellenálló hordozható gépeiről híres Panasonic CF-18 jelzésű Toughbook modellje több újdonságot is felvonultat. Ez az első olyan mostoha körülmények közé szánt, különlegesen megerősített TabletPC, ami egyszerre képes vezeték nélküli LAN, Bluetooth, CDMA és GPS kapcsolatot teremteni. A gép katonai minősítéssel is rendelkezik, bírja a rázkódást, az ütődést, a vizet, a port, a hőséget és a hideget; kijelzője 850 cd/m² fényerővel bír, ami szabad téren, erős napsütésnél is lehetővé teszi a használatát. Tömege nagyjából 2 kg, akkumulátora 7,5 órás üzemidőt biztosít.

Rance Poehler, a Panasonic Computer Solutions Company elnöke szerint a normál hordozható gépek meghibásodási aránya húsz százalék feletti, és véleménye szerint az átlagos tábla PC-k ennél is nagyobb arányban fognak elromlani, ha szélesebb körben elterjednek – és ekkor még szó sincs arról, hogy ezeket a gépeket a tulajdonosaik szabad térben használnák. Azt elfelejtette hozzátenni, hogy a legújabb Toughbook 3200 dolláros árából akár három normál gépet is kapni lehetne, így – ha csupán a gépvásárlás költségét nézzük, tehát a kényelmetlenséget, az esetleges adatvesztést és a kiesett munkaidőt nem vesszük figyelembe – a hagyományos eszközök mellett döntő felhasználónak bőven lesz mit összetörnie.

Közel-Keletre készülve

A háború mindig nagy üzlet – kérdés, kinek –, miért lenne ez másképp a számítástechnika világában?

A Lind Electronics kifejezetten a Közel-Keletre készülődő tudósítóknak Apple Titanium G4 hordozható gépekhez ajánlja Lind Embedded Reporter Package nevű csomagját, ami egy sok mindent kibíró akkumulátortöltőt és egy 8–14 órás használati időt nyújtó külső akkumulátort tartalmaz. A 700 dolláros csomag 12 és 28 voltos áramforrást kínáló járművekben egyaránt használható, csatlakozója NATO-szabvány szerinti; mint a cég közleményéből megtudni, a HUMVEE terepjárókban például elég az anyósülés alá benyúlni, hogy megfelelő áramforráshoz jussunk. Azt, hogy a szövetségesi UAZ járművekben hol kell a megfelelő aljzatot keresni, nem közölték.

➲ http://www.lindelectronics.com

Névvita sokadszorra

A Microsoft 2001 decemberében panaszolta be a linuxos alapú Windows-imitátor operációs rendszer készítőjét, mivel az – véleménye szerint – megsértette a Windows név mint bejegyzett védjegy használatát. A Lindows.com érvelése szerint a Windows az ablakok angol megfelelőjeként túlságosan általános megnevezés ahhoz, hogy védjegyként használhassák.
A Seattle-i bíróság most arra kötelezte a Microsoftot, hogy mutassa be a korábban az Apple–Microsoft-pereskedés során született iratokat. Akkor ugyanis a Microsoft sikeresen érvelt amellett, hogy a grafikus felhasználói felület, az ablakok, az ikonok és egyéb kellékek fejlesztése az 1970-es években több cégnél is megkezdődött, így nem válhatnak egyetlen társaság tulajdonává, és nem is védhetők le. Most hasonló gondolatmenet vezethet arra a következtetésre, hogy a Windows mint általánosan elterjedt, a szótárakban és a lexikonokban is – nem csak a Windows megjelenése óta – szereplő név, nem lehet egyetlen cég tulajdona, így a hozzá fűződő jogokat sem lehet megsérteni. Emellett már az is kérdéses, hogy a Microsoft ilyen adottságok mellett hogyan védethette le a nevet, amikor annak bejegyzését éppen általánossága miatt az illetékes hivatalok többször is megtagadták.
A következő tárgyalás időpontja december 1-je – akkor talán kiderül, hogy a világ egyik legismertebb és legértékesebb elnevezésének mi lesz a sorsa.

Grandiózus tervek

Az Intel Developer Forumon érdekes tudnivalók kerültek napvilágra az Intel következő nemzedékbeli Pentium 4 processzorával kapcsolatban. A Prescott lapka 90 nm-es eljárással készül majd, és órajelét tekintve egészen 5 GHz-ig nyújtózkodik. Az Intel mérnökei azt is elárulták, hogy egy újfajta órajel-továbbítási módszerrel jobban szinkronban tudják tartani a lapka részeit – ezt a megoldást egyébként más termékekben is alkalmazni kívánják. A 13 új utasítást is ismerő Prescott lesz az Intel első olyan processzora, amelyik támogatja a Le Grande kódnevű megoldást. Ugyan részleteket nem közöltek, de a Le Grande várhatóan egy védett területet fog fenntartani a számítógép központi memóriájában, ahol biztonságosan futhat a Microsoft által fejlesztett, számos érdekelt cég által támogatott, ám a felhasználók által legalábbis kétkedve (nem) várt szerzői jogvédelmi eszköz, a Palladium.

Hűsítő HP

A HP új, az adatközpontok hűtésének tervezésére szolgáló megoldása a cég állítása szerint évente akár több millió dolláros költségmegtakarítást is hozhat ügyfelei számára. A levegő áramlását számítógép segítségével – a repülőgépek tervezése során használatos módszerhez hasonló – szimuláló megoldás elkészíti az adatközpont 3D-s modelljét, amelyen megvizsgálható a hőeloszlás és a levegő áramlása, majd ennek alapján javaslatot tesz az eszközök és légkondicionáló berendezések elhelyezésére. Hatékony elrendezést alakít ki, hogy a hűtéshez felhasznált energia mennyisége jelentősen, nagyobb géptermek esetében akár 25 százalékkal is csökkenthető legyen.
A szolgáltatás fejlesztésével a HP megoldása a későbbiekben arra is képes lesz, hogy a berendezések indítását és leállását figyelembe véve dinamikusan módosítsa a hűtést, illetve a lehető legtakarékosabb légmozgatásra törekedve különböző adatközpontok között osszon el bizonyos feladatokat.

eBook német és francia nyelven

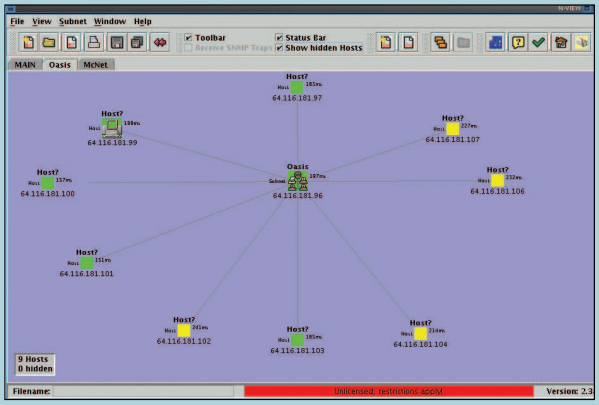
A Palm Digital Media – a PalmSource vállalkozása – bejelentette, hogy két cég Európában is Palm Reader eBook-kiadványok forgalmazásába kezdett.
A berlini envi.con KG és a párizsi Giant-Chair többek között a Bertelsmann és a Hachette könyveit fogja forgalmazni, német és francia nyelven. Az előbbi kínálatában többek között ***Peter Scholl-Latour***, ***Richard von Weizsacker***, ***Felix Huby***, ***Franz Kafka*** és ***Edgar Allen Poe*** művei szerepelnek, míg francia nyelven számítástechnikai és gazdasági kiadványok, különféle regények lesznek elérhetők, illetve a Le Monde bizonyos cikkei.
A Palm Digital Media kínálatában jelenleg mintegy tízezer elektronikus kiadvány szerepel. A Palm Reader program használója viszonylag szabadon válogathat, hogy milyen eszközön szeretne olvasgatni; a program Palm OS és Pocket PC alapú, illetve windowsos és Macintosh asztali gépekre egyaránt elérhető.
➲ http://ebooks.pdassi.de
➲ http://www.palmlivres.com

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 hírszerkesztője.

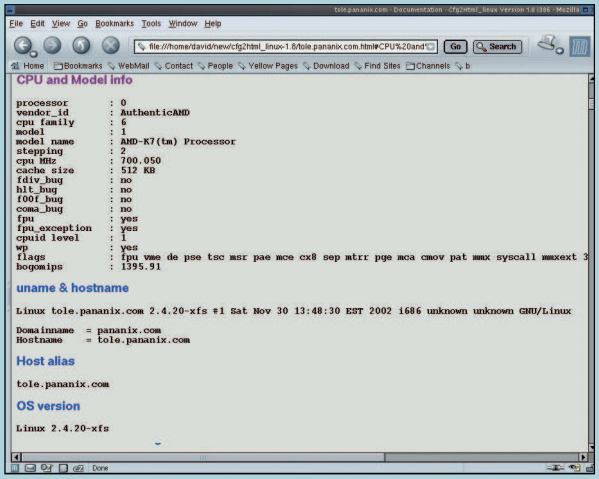
Falatkák

N-View

A `tkined` felhasználói számára nem lesz idegen az N-View hálózatfelügyeleti program. Az N-View azonban jóval gyorsabb és könnyebben használható. Több képernyő beállítható benne a



különböző alhálózatok számára. Legnagyobb hátránya – nevezetesen az, hogy JRE szükséges hozzá – egyszerűen megoldható, ha JRE-t tartalmazó csomagot használsz. Az N-View levélben



értesít, ha valamelyik rendszer elérhetetlenné válik, bár ennek nem sok hasznát látom, ha az adott rendszer éppen a levélkiszolgáló. Futtatásához Java szükséges.

➲ http://www.n-view.de/index_en.html

cfg2html

Ez a segédprogram végigpásztázza a rendszert, és rengeteg adatot rögzít róla, amelyek alapján szinte egy az egyben újraépítheted a rendszert. Ennyire részletes adatokat szerintem nem célszerű a weboldalra kitenni, ellenben igen hasznos kinyomtatni és a rendszer leírásához hozzáfűzni (mert ugyebár van rendszerleírásod?!). Futtatásához `bash` és szabványos unixos eszközök szükségesek.

➲ http://members.tripod.com/rose_swe/cfg/cfg.html

David A. Bandel

Linux Journal 2003. március, 107. szám

Cégvilág

Negyedével csökkenő internetárak?

Az IHM és a Vivendi 2003. március 14-én aláírt egy megállapodást, miszerint Az Európa Terv programja szerint márciustól átlagosan 25 százalékkal csökkennek az internet-előfizetési díjak. Ennek keretében újabb kétoldalú megállapodás aláírására került sor *Kovács Kálmán*, informatikai és hírközlési miniszter és *Polányi Sándor*, a Vivendi Telecom Hungary vezérigazgatója között. A minisztérium a vezetékes telefonszolgáltatóval a tavaly augusztusban megkötött támogatási szerződést módosította. A Vivendi által az internetszolgáltatók számára nyújtott nagykereskedelmi konstrukcióinak köszönhetően az internetszolgáltatók maguk alakíthatnak ki kedvezőbb árú internetcsomagokat. Ezzel megteremtődött a lehetősége annak, hogy az ország szinte teljes területén 25 százalékkal csökkenjen az internethasználat költsége.

A HP és az életmentés

Az egészségügyben sokszor másodperceken múlhat egy-egy beteg élete, az orvosok nem hibázhatnak – ehhez azonban elengedhetetlen, hogy mindig hozzáférjenek a beteg egészségügyi irataihoz (korábbi leletek, érzékeny-e valamilyen gyógyszerre stb.). A Los Angeles-i Kaliforniai Egyetem (UCLA) orvosi központjának szakemberei évente egymillió esetet látnak el a sürgősségi osztályon, több mint húszmillió új iratot készítenek a betegekről.

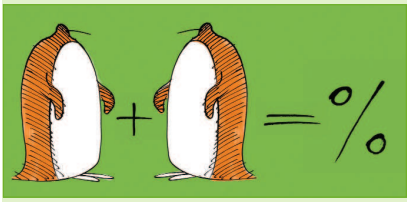
A kórház az iratokat feldolgozza és tárolja, hiszen életbevágóan fontos, hogy szükség esetén minden dokumentum kéznél legyen. Mivel az iratok feldolgozása és dokumentálása sok időt, valamint emberi és pénzügyi erőforrás igényel, a UCLA orvosi központja a HP-hez fordult segítségért.

A HP receptje: két Digital Sender 9100C dokumentumkezelő rendszer. Az orvosi központ informatikusai alig egy hónap alatt dolgozták ki azt a megoldást, amelynek segítségével a Digital Senderek beolvassák a betegek ellátásával kapcsolatos dokumentumokat, majd azokat egy biztonságos webkiszolgálón tárolják. Az orvosok csak behelyezik az oldalakat a lapadagolóba, megnyomják az „orvosi dokumentumok” billentyűt, és a HP Digital Sender 9100C az iratokat pillanatok alatt biztonságosan hozzáférhető digitális fájlokká alakítja.

➲ http://www.hp.hu

Linux-index

1. Az 1927 és 1946 között megjelent mozifilmek ennyi százaléka nem szerezhető be jelenleg: 93
2. Ennyi kormányhivatalban használt asztali gépen tértek át Linuxra a spanyol Extremadura tartományban 2002 novemberéig: 10 000



3. Várhatóan ennyi kormányhivatalban használt asztali gépen térnek át Linuxra a spanyol Extremadura tartományban 2003 novemberéig: 100 000
4. Ennyi példányban töltötték le Extremadura saját Linux-változatát, a Linexet a tartományon kívül: 55 000
5. Ennyi tucat országban támogatja törvény a szabad programok használatát: 2
6. Ezekben az országokban a szabad programokkal kapcsolatos, elfogadásra váró törvénytervezetek és jogszabályok száma: 70
7. A linuxos rendszerkezelési eszközök száma az IBM Tivoli termékében 2001-ben: 2
8. A linuxos rendszerkezelési eszközök száma az IBM Tivoli termékében 2002-ben: 20
9. Az informatikai vezetők ennyi százaléka használ „bizonyos mértékben” Linuxot: 39
10. A különféle Linux alapú kézi gépek száma: 23
11. Ennyi kiszolgáló található a buffalói egyetem egyik Linux-telepében, amelyet 2002 szeptemberében telepítettek: 2 000
12. Ennyi kiszolgáló található a buffalói egyetem egy másik Linux-telepében, amelyet 2002 novemberében telepítettek: 300

Források
1.: *Jason Schultz*naj *Lawrence Lessig*-hez intézett levelében
2–6.: Washington Post
7–8.: Information Week
9.: Goldman Sachs Research, IDGnet
10.: LinuxDevices.com
11.: Boston Globe

Új LME-csoportok

A Linux-felhasználók Magyarországi Egyesületében (LME) megalakult a Debian (LDCs) és a WebTeam csoport.

Az LME Debian csoport célkitűzései

- A magyarországi Debian GNU/Linuxot használóknak és fejlesztőknek szervezeti formát nyújtani.
- A hazai Debian GNU/Linuxszal kapcsolatos kezdeményezések felkutatása, elindítása, megszervezése és támogatása az elérhető anyagi és emberi erőforrások révén.
- Elősegíteni a Debian GNU/Linux minél szélesebb körben való elismertségét, természetesen nem az egyéb Linux-terjesztések kárára.
- Az LDCS közösségi életének megszervezése (például rendszeres találkozók rendezése).
- Az LME Debian levelezőlista üzemeltetése.
- Kapcsolatfelvétel és kapcsolattartás a hazai és külföldi társszervezetekkel és társcsoportokkal.
- A csoport jó hírnevének kialakítása és képviselete hazai és külföldi rendezvényeken.

A csoport elnöke: *Holányi János* (Csan) <csani@lme.linux.hu>
A csoport honlapja a ➲ http://www.lme.hu/debian/ címen található.
Levelezőlistájuk a debian@lists.linux.hu címen érhető el.
A levelezőlistára feliratkozni a következő címen lehet:
http://lists.linux.hu/mailman/listinfo/debian

WebTeam
A WebTeam projekt az egyesület weboldalainak üzemeltetésével és karbantartásával foglalkozik. Ezek a linux.hu és az lme.hu oldalak.
A csoport vezetője: *Kovács Zsolt* <zsolt_k@freemail.hu>

Gibizer Tibor (gibzo@linuxmania.hu)

Visszatekintés

A három évvel ezelőtti ismertetett programok közül szemezgetve remek segédprogramokra bukkantam. Szerepelt köztük a Downloader for X, amit rendszeresen használok, valamint a `gnotepad+` és az `xglobe`. Végül a Note győzött. A Note semmi különlegeset nem nyújt azon kívül, amit a neve (jegyzet) ígér, azaz jegyzetkészítésre használható. A jegyzeteket bináris, DBM vagy a Perl által támogatott bármilyen más SQL-formátumban tárolja. A feljegyzések szerkeszthetők, törölhetők, illetve témakörök szerint könyvtárakba csoportosíthatók. A jegyzetadatbázis tartalmazhat egyszerű szöveget, vagy akár titkosított is lehet, így kiválóan alkalmas jelszavak és egyéb bizalmas adatok tárolására. A futtatáshoz szükséges: Perl, Perl modulok: IO::Seekable, DBI::mysql (vagy más DBI-függő modul, választható), DB_File (választható), MD5 (választható), Crypt::IDEA (választható), Crypt::DES (választható), Crypt::CBC (választható).
➲ http://www.0x49.org

David A. Bandel (dbandel@pananix.com)

Linux Journal 2003. március, 107. szám

diff -u: rendszermag-fejlesztési hírek

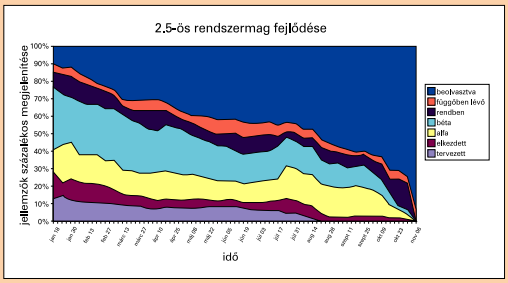
Nem sokkal azután, hogy múlt október végén befagyasztották a rendszermagot, *Guillaume Boissiere* elhatározta, hogy kimutatást készít a 2.5-ös fa szolgáltatásbővítéséről. Áttanulmányozta a 2.5-ös fejlesztés szinte teljes történetét a 2002. év elejétől kezdve. Állapotuk szerint a következő hét csoportba sorolta a szolgáltatásokat: tervezés, elkezdődött, alfa, béta, beépíthető, beépítés felfüggesztve és teljesen beépített. Az első fejlődési ábrán látható, hogy a szolgáltatások milyen arányban tartoztak az egyes csoportokba. A második fejlődési ábra azt mutatja, hogy ténylegesen hány szolgáltatás került egy-egy csoportba az idő előrehaladtával. Bár teljes pontosságot ezektől az ábráktól nem várhatunk, nagyon érdekesek – ha másért nem, azért mindenképp, mert leolvasható róluk, hogy mennyire vette komolyan a fejlesztők többsége a befagyasztás határidejét. Megfigyelhető, hogy a fejlesztés a nyár folyamán felgyorsult, majd az új szolgáltatások tervezésének idejére teljesen lecsengett. A tevékenység fokozódása nagyjából arra az időre tehető, amikor megszületett a döntés a rendszermagbővítés novemberi befagyasztásáról.

A szolgáltatásbővítés lezárásától a 2.6-os (vagy 3.0-s) változat tényleges kiadásáig tartó szakasz elősegítésére az Open Source Development Lab (OSDL) munkaerőt és berendezést biztosított a Linux-rendszermagfejlesztésnek egy Bugzilla hibakövetési adatbázis karbantartására a ➲ http://bugzilla.kernel.org webhelyen. A fejlesztők kezdetben határozottan támogatták ezt a nyilvántartást, ám a lelkesedés kissé lelohadt, amikor *David S. Miller* és más nagymenők ismétlődő bejegyzéseket találtak, és a komolytalan jelentések miatt a rendszer – legalábbis az eredeti formájában – inkább zavarossá vált, semmint hasznossá. Mindazonáltal senki sem akart felhagyni vele, s úgy tűnik, jelenleg közös erőfeszítéseket tesznek annak érdekében, hogy a hibaadatbázis használható legyen.

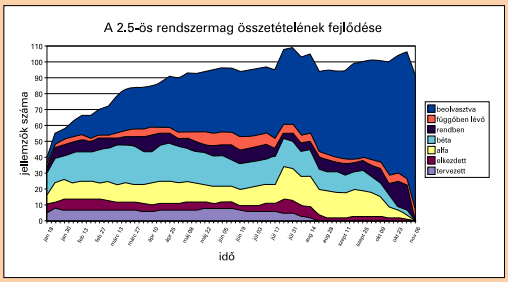
Linus Torvalds először mutatott hajlandóságot arra a nyomkövetés kapcsán, hogy javítófájlokat fogadjon el egy rendszermagalapú hibakereső programhoz. Linus korábban azt az álláspontot képviselte, hogy az igazi programozók a forrásfájlokat használják hibakereséshez. Habár magyarázatot nem adott rá, hogy miért változtatott a véleményén, most úgy tűnik, jó ötletnek tartja egy hálózaton keresztül futó rendszermag-hibakereső program bevételét a rendszermagba. A következő üzembiztos kiadásban azonban még kár keresni, mivel Linus – igen óvatosan – elhatározását csak a rendszermagbővítés lezárása után jelentette be.

Október végén új, a cramFS-hez hasonló, csak olvasható, tömörített fájlrendszer tűnt fel és vette célba a 2.4-es rendszermagot. A SquashFS állítólag gyorsabb és jobban tömörít, mint a zisoFS vagy a cramFS. Szerzője, *Phillip Lougher* a többi tömörített fájlrendszer korlátozásaira keresett megoldást, különösen a legnagyobb fájlméretet, a fájlrendszerméretet, illetve a blokkméretet illetően.

Ha már fájlrendszerekről beszélünk, emlékszik még valaki a xiaFS-re? 1993-ban az `ext2`-vel együtt komoly esélye volt rá, hogy a legelterjedtebb fájlrendszerré



1. ábra
Szolgáltatások megoszlása az egyes csoportok között



2. ábra
Tényleges szolgáltatásszám az egyes csoportokban

váljon. A két versenytársat mindössze pár hét különbséggel kezdték nyilvánosan használni, s egy ideig úgy tűnt, hogy a xiaFS veszi át a vezetést. 1994-re azonban jelentősen visszaesett a népszerűsége, s pár év múlva törölték is a hivatalos rendszermagfából. 2000-ben Linus felvetette, hogy mókás lenne visszavenni. Végül közvetlenül a novemberi befagyasztást követően *Carl-Daniel Hailfinger* megkérdezte, hogy még áll-e ez az ajánlat. Linus azt mondta, hogyne, sőt hajlandó kivételt tenni vele a szolgáltatásbővítés lezárását illetően, amennyiben Carl hozza a javítócsomagot.

Zack Brown

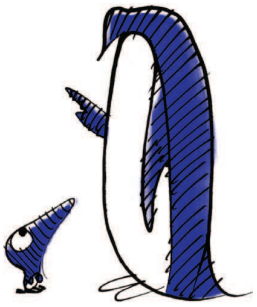
Linux Journal 2003. március, 107. szám

Ők mondták

Eddig számos kormányhivatalt látogattam meg. Gyakorlatilag az összes kormányzati hivatalban, ahol jártam, lehet Linuxot találni…
A kérdés az, hogy vajon tud-e róla valaki? Gyanítom, hogy azoknak, akik saját állításuk szerint nem használják, legalább a fele alkalmazza, csak éppen nem tud róla. *(Robert Hibbard – Red Hat)*

Még nem fejlődött ki a megfelelő szókincs, amely helyesen fejezné ki a nyílt forrás dinamikáját, nem pedig rejtélyes, elfajult betyárkodásnak tüntetné fel. Ha majd meglesz a szókincs – a minőség–ár összehasonlításának módszere –, az őt megillető csúcsra emeljük a nyílt forrást: a legtermelékenyebb folyamat egy, a termelékenységet mindenek elé helyező gazdaságban. *(Britt Blaser)*

Linux Journal 2003. március, 107. szám



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

A hónap szakmai tanácsai

Melyik GUI-eszközkészletet válasszam?

Olyan programot írok, ami laboratóriumi kísérleteket vezérel és elemzi az adatokat. Tetszőlegesen hosszú RPN stílusú parancsokat fogad el. A parancsok visszaidézése és szerkesztése, amit például a readline biztosít, létfontosságú, és a programnak grafikus képernyőn kell futnia. Az svgalib használata felé hajlok, mert ez megkönnyítené az áttérést a DOS-os változatról, és nagyobb hatékonyságot nyújtana a grafikonok rajzolásában, de más megvalósítások lehetőségét is meg fogom fontolni. Tudnátok néhány általános ötletet adni a parancssor megvalósítására?
Bill McConnaughey, mcconnau@biochem.wustl.edu

Attól függően, hogy melyik kinézet, illetve programnyelv a legkényelmesebb neked, talán akkor jársz a legjobban, ha egy modern GUI-eszközkészletet választasz, például a GTK+-t vagy a Qt-t. Sokkal több kódot tudsz belőlük átvenni. Hosszú távon is könnyebb lesz a dolgod, ha a képernyők méretének növekedésével a programod mellett egyszer egy másik programot is szeretnél ugyanazon a képernyőn futtatni.
Ricardo Fernández Pascual egy érdekesnek tűnő önműködő kiegészítésképességet írt a GtkEntry elemhez. Projektjének a neve EggEntry, és elég általánosnak tűnik ahhoz, hogy bonyolult parancsok bevitelét támogassa:
➲ http://www.geocrawler.com/mail/msg.php3?msg_id=9808742&list=521.
Don Marti, dmarti@ssc.com

A kettős rendszer telepítéskor lefagy

Mostanában vettem egy Compaq Presario 1516US számítógépet, amelyhez Microsoft Windows XP járt. A Partition Magic segítségével felosztottam a merevlemezt. Miután behelyeztem a Red Hat 8.0 telepítő CD-jét, a telepítés átment a kezdő ellenőrző lépéseken. Amikor a következő lépéshez ért, lefagyott:

```
Partition Check:
 hda:
```

A kurzor villog, és semmi sem történik.
Avran, idontlikemail@earthlink.net

Az alábbi oldalon egy jó oktatóanyagot találsz a Linux GRUB rendszerbetöltőjének telepítéséről két operációs rendszert használó számítógépre, amelyen a Windows a Linux telepítése előtt már telepítve volt:
➲ http://www.geocities.com/epark/linux/grub-w2k-HOWTO.html.
Felipe E. Barousse Boué, fbarousse@piensa.com

Egyetlen lemezfelosztást változtató eszköz sem ad százszázalékos biztonságot. Mindegyik figyelmeztet, hogy először készítsünk biztonsági mentést. A két operációs rendszer használata a munkavégzés szempontjából nem hatékony, mert az éppen szükséges alkalmazás látszólag mindig a másik operációs rendszer alatt van telepítve. Ha már mindenképpen a kettős rendszer mellett döntesz, készíts biztonsági mentést az eredeti operációs rendszerről, hogy szükség esetén visszaállíthasd az eredeti állapotot.
Don Marti, dmarti@ssc.com

A Modem Lights kisalkalmazás nem hagy kilépni

Red Hat 8.0 környezetben a Modem Lights kisalkalmazást használom a PPP-kapcsolat felépítésére, de nem tudom alkalmazni a szétkapcsolásra. Ha ismét megnyomom a gombot, hogy kilépjek, a program még egyszer megkérdezi: *Do you want to connect?* Ha nemleges választ adok, nem történik semmi. Ha igenlő választ adok, a program szétkapcsol, majd ismételten csatlakozik.
Martin A. Boegelund, goblin@linuxmail.org

Az alapértelmezett beállításokkal van itt a gond. A Modem Lights kisalkalmazás *Preferences* párbeszédablakában menj az *Advanced* fülre. Itt állítsd be a modem zárolófájlját. Próbáld meg erre beállítani: */var/lock/LCK.modem*.
Ben Ford, ben@kalifornia.com

Felügyelet monitor és billentyűzet nélkül

Olyan x86 alapú keretbe építhető kiszolgálókat keresek, amelyek monitor és billentyűzet nélkül kezelhetők. Jelenleg több olyan Sun V100-as rendszert használok, amelyek képesek erre, de itt a Solaris használatára kényszerülök. A kaputovábbítás módszerével már sikerült „fej nélküli" rendszereket telepítenem a soros csatolóra, de ez nem oldja meg azokat a gondokat, amelyek az operációs rendszer lefagyásakor lépnek fel, amikor ki- és be kellene kapcsolni a gépet, vagy az operációs rendszert kellene a távolból frissíteni.
Ron Culler, ron@firelan.net

Egy gépfelügyelő kártyára van szükséged. Sok nagy Intel alapú kiszolgálókat gyártó cég, többek között a Dell, a HP és az IBM ajánl a kiszolgálóihoz ilyen kártyákat választható összetevőként. E kártyák működése gyakorlatilag a konzolnak a soros kapura történő átirányításához hasonlít, kivéve hogy más dolgokra is használhatóak, például a rendszernek távolból való újraindítására vagy a kiszolgáló teljes kikapcsolására. Néhány ilyen kártya még hálózati kapuval is rendelkezik, kiküszöbölve a soros alapú kommunikációt.
Chad Robinson, crobinson@rfgonline.com

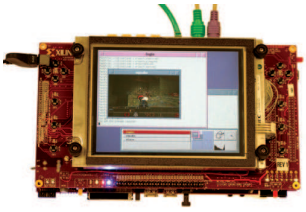
Néhány alaplap, például az Intel 440GX egy második soros kapuval rendelkezik, amely pont arra használható, amire neked szükséged van (a gép megfigyelésére, újraindításra és kikapcsolásra). Természetesen a BIOS is átirányítható erre a kapura. Az EMP (vészkezelő kapu) vezérléséhez Linux-környezetben a VACM-et használhatod, lásd ➲ http://vacm.sf.net.
Marc Merlin, marc_bts@google.com

A Middle Digital, Inc. gyárt olyan ISA és PCI kártyákat, amelyek a legtöbb PC alapú rendszert „fej nélküli" képességgel ruházzák fel. Telnetelj be a demo.realweasel.com címre, és nézd meg élőben, vagy keress rá a „weasel" kifejezésre a Linux Journal honlapján.
Don Marti, dmarti@ssc.com

Új termékek

Xilinx Virtex-II Pro ML300 Evaluation Platform

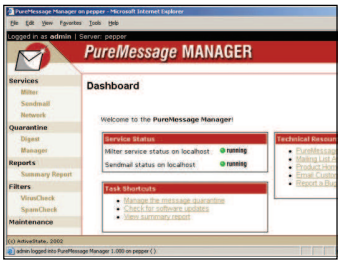
A Xilinx Virtex-II Pro ML300 Evaluation Platformja a fejlesztők számára lehetővé teszi, hogy megismerjék a beágyazott PowerPC processzort tartalmazó Virtex-II Pro FPGA tulajdonságait. Ez egy olyan fejlesztőfelület, ami a PowerPC-t, a RocketIO adóvevőt és más Virtex-II Pro FPGA tulajdonságokat kihasználó termékek tervezésére való. A termék tartalmazza a GNU-eszközöket és -referenciaterveket, a ChipScope Pro 5.li hibakereső eszközt, több mint negyven paraméterezhető IP-alapmagot (core), a Xilinx ISE 5.li FPGA megvalósító eszközök próbaváltozatát és a vezetékeket. Az ML300 MontaVista Linux Professional Edition környezetben támogatott.
Az alaplap tulajdonságai: négy gigabites ethernetkapu, két soros ATA-



csatlakozó, két HSSDC2 csatlakozó, SystemACE CF felület 1 GB-os IBM MicroDrive-val, 128 MB DDR SDRAM, 6,4" VGA TFT LCD beépített érintőképernyő-képességgel, IEEE-1394 támogatás, 32/33 PCI Mezzanine Card bővítőhely, két CardBus bővítőhely, AC97 Audio, IIC EEPROM, hőmérsékletérzékelők, digitális potenciométerek; SPI EEPROM; PS2, soros és párhuzamos kapuk, valamint egy 62 négyzet-centiméteres prototípusterület 90 szabad FPGA B/K-val.
Adatok: Xilinx, Inc., 2100 Logic Drive, San Jose, California 95124,
telefon: 408-559-7778,
➲ http://www.xilinx.com/ml300

PureMessage 3.0

A PureMessage (korábban PerlMx) bejelentette reklámlevél-ellenes, vírusirtó és irányelv-betartató termékének 3.0-s változatát. A program gátat szab a nem kívánt elektronikus leveleknek és védi a végfelhasználók postafiókját. A PureMessage nyílt forrású programokon alapul, például a SpamAssassinon és a WebAdminon; Perl nyelven bővíthető, és a legtöbb Unix-felületet támogatja.
A 3.0-s változat a következőket tartalmazza: webalapú kezelőfelület az irányelvek beállításához, továbbfejlesztett levélszemét-felismerő, rugalmas kezelés, választható végfelhasználói karanténkezelés és



McAfee vírusirtómotor. A PureMessage levélszemét-felismerő képességét a levélszemét-ellenes elméletek fejlődésével összhangban rendszeresen frissítik.
Adatok: ActiveState Corporation, #400 - 580 Granville Street, Vancouver, BC V6C 1W6, Canada,
telefon: 604-484-6400,
e-mail: support@activestate.com,
➲ http://www.activestate.com

Fonix DECtalk RT

A DECtalk szövegfelolvasó program egy formánsalapú beszédszintetizátor, amely a közönséges szöveget természetes hangzású, érthető beszéddé alakítja. A DECtalk személyre szabható hangszíneket kínál, és a részletekbe menő beállítási lehetőségek biztosítják a jó teljesítményt a mindennapi alkalmazásokban.
A Fonix DECtalk RT for Linux többnyelvű szövegfelolvasást tesz lehetővé; kilenc előre meghatározott hangszín érhető el; ki tud ejteni hangokat; szavakat és kifejezéseket; a beszéd szüneteltethető, folytatható és leállítható; tetszőleges szövegfájlba lehetséges a beszédet vezérlő parancsokat beépíteni stb. A DECtalk letölthető, és futásidejű összetevőként együttműködik minden olyan alkalmazással, amelyet erre felkészítettek.
Adatok: Fonix, 180 W Election Road, Draper, Utah 84020,
telefon: 801-553-6600,
➲ http://www.fonix.com

Powerwall 3

Az AdminForce Powerwall 3 Network Security System (3NSS) kis- és közepes vállalkozások számára nyújt biztonságellenőrzési és behatolásérzékelési szolgáltatásokat. A Powerwall 3NSS távolból telepíthető, figyelhető, hetente jelentést készít az eseményekről, riaszt és idejében figyelmeztet. A proxyszolgáltatások a Powerwallba be vannak építve, beleértve a következőket: DNS, web, e-mail, CUSeeMe, RealAudio és RealVideo. A VPN szolgáltatásengedélyezés bebocsátja vagy letiltja bármilyen TPC vagy UDP alapú alkalmazás hozzáférését. A Powerwall 3 egy 19"-es keretbe szerelhető házban kapott helyet. Megtalálható benne 128 MB RAM, legfeljebb négy ethernetkapu, Token Ring és FDDI-támogatás, parancssoros felület és címálcázás.
Adatok: AdminForce Remote LLC, 240 Copley Road, Upper Darby, Pennsylvania 19082,
telefon: 610-734-1900,
e-mail: sales@adminforce.net,
➲ http://www.adminforce.net

CrossOver Office Server Edition

A CodeWeavers megjelentette a CrossOver Office Server Editiont, ami a Linuxot vagy Solarist futtató osztott vékony ügyfeleket használó nagyvállalati felhasználó számára lehetővé teszi, hogy MS Windows programokat használjanak anélkül, hogy Microsoft operációs rendszerrel és a hozzá tartozó engedélyekkel rendelkeznének. A CrossOver Office támogatja az irodai alapcsomagokat, például az MS Office-t, az Outlookot, az Internet Explorert és sok más üzleti alkalmazást. A Server Edition az egy idejű felhasználók mennyiségétől függően árazza engedélyeit, de korlátlan számú felhasználót engedélyező vállalati engedély is elérhető.
Adatok: CodeWeavers, Inc., 2550 University Avenue West, Suite 439S, St. Paul, Minnesota 55114,
e-mail: sales@codeweavers.com,
➲ http://www.codeweavers.com

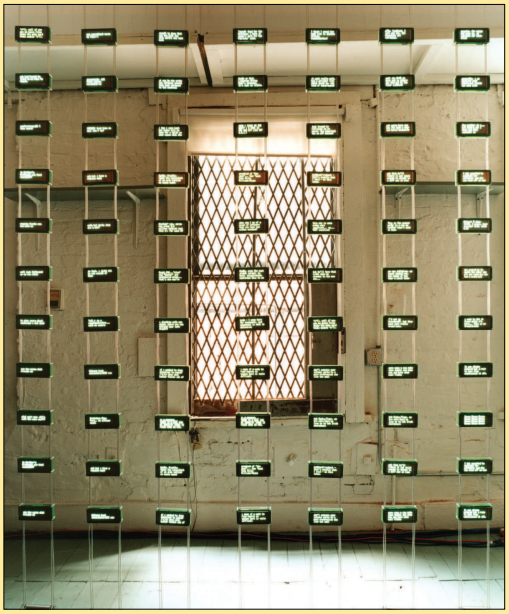
Linux Journal 2003., 107. szám

A Listening Post: hálózati fórumok a művészetben

Kihagyhatatlan élményt nyújt a Listening Post című installáció. *Mark Hansen* és *Ben Rubin* alkotása jelenleg a New york-i Whitney Modern Művészeti Múzeum kiállításán látható (➲ http://www.whitney.org). Ez a rendkívüli darab, ami egyszerre négy számítógépen és ugyanennyi különböző operációs rendszeren fut (természetesen ezek egyike a Linux), az Internet egyesített hangját fejezi ki, multimédiás berendezéssé alakítva át a hálózati kapcsolattartást.

Az alkotás honlapján olvasható ismertető szerint a Listening Post „statisztikai elemzéssel az üzeneteket tartalmuk szerint témakörökbe rendezi, nyomon követve a hálózaton folyó kapcsolattartás hullámzását. A kimondott szót hangok játéka festi alá, melynek magasság- és hangszínbeli változásai az üzenetek áramlásának és tartalmának váltakozását követik."

2002 novemberében seattle-i tartózkodásom alatt volt szerencsém „megtekinteni" a Listening Postot. Elsőre az alkotásból áradó szinte kísértetiesen emberi érzet ragadott meg – szokatlan, hogy a számítástechnika efféle költőiséggel társuljon. Egy elsötétített szobában, párnákkal a földön, falnyi apró képernyő sugározza zöld betűkkel a több ezer nyilvános hálózati csatornáról valós időben gyűjtött szöveget. A szövegtöredékeket egy számítógép által létrehozott, brit kiejtéssel beszélő hang kíséri, amely felolvas egy-egy véletlenszerűen kiválasztottat a villogó üzenetek közül. Különösen nagy hatással volt rám a „... vagyok" sorozat; „... vagyok" formájú, valós idejű üzenetek hangzanak el a sötétben: „Fáradt vagyok." „Boldog vagyok." „Norvég vagyok." Száz meg száz ember, akik éppen abban a pillanatban írják le önmaguk legalapvetőbb vonásait, ki tudja, honnan vagy kinek, míg képzeletem száguldva próbálja meg kiegészíteni történeteiket. Nehéz találóan visszaadni a Listening Post tiszavirágéltű természetét; a mű legjobb jellemzésével a két alkotó egyike, Ben Rubin szolgál művészi nyilatkozatában: „Mindenki, aki begépel és elküld egy üzenetet egy csevegőszobában, valamiféle válaszért kiált. A Listening Post a mi válaszunk – a szövegek látványos elrendezéseinek és zenei aláfestések egy sorozata, ami ennek az üzenetáradatnak a méretére, azonnali jellegére és a tartalmára válaszol. Minden egyes szót, amely bekerül a rendszerünkbe, csak néhány másodperccel előbb gépelte be valahol valaki.



A beérkező üzenetek szaggatott üteme adja a mű látható és hallható ritmikáját. A hangképző rendszerek a szélcsengőkhöz hasonló felépítésűek, csak itt a szél nem meteorológiai, hanem emberi eredetű, és a mozgó részecskék nem levegőmolekulák, hanem szavak. Bizonyos értelemben a Listening Post lényege az, hogy befogja az emberi energiát, amit ez a sok szó hordoz, és átvezesse azt a rendszer gépezetén.

A Listening Post képviseli Mark Hansennel való együttműködésem legjelentősebb eredményét, aki az általam ismert művészek közül az egyetlen, akinek a statisztika a kifejezőeszköze. Mióta együtt dolgozunk, fogalmi szókincsem rengeteg olyan kifejezéssel bővült, ami Mark mindennapi munkájának szerves része, például telepépítés, simítás, kiugróan eltérő adat, nagy dimenziószámú tér, valószínűségi eloszlás stb. Mark szemszögéből pillantván meg a világot olyan hangokra lettem figyelmes, amelyeket korábban eszembe sem jutott volna keresni."

Kiállítási dátumokat és egyéb adatokat a Listening Post honlapján találunk.

➲ http://www.earstudio.com/projects/ListeningPost.html

Jill Franklin

Linux Journal 2003. március, 107. szám

Minden egyes szót, amely bekerül a rendszerünkbe, csak néhány másodperccel előbb gépelte be valahol valaki.

Újabb MPAA támadás: műsorszórási fejléc

Betiltja-e az amerikai kormány a szabad meghajtók fejlesztését a tévévevőkhöz?

A „Digitális Évezred Szerzői Jogi Törvénye” (Digital Millennium Copiright Act, azaz DMCA) jelentősen kiterjesztette a szerzői jogok birtokosainak a jogait a jogdíjas protokollok és médiaformátumok megvalósítási módjainak ellenőrzésére. Ugyanakkor lényegében érintetlenül hagyta a nem jogdíjas formátumok szabályozását. A DMCA értelmében még szabadon valósíthatjuk meg nyílt szabványokra épülő rendszerünket, olyan tulajdonságokat építve bele, amiket jónak látunk.

Most azonban még ez az „előírás nélküli” szabályozás is támadások kereszttüzébe került. Olyan javaslatok jelentek meg, amelyek a nyílt szabványok felhasználásának szabályozását tűzik ki célul. A legismertebb a Digitális Műsorszórást Szabályozó Törvény (CBDTPA), amelyet ***Ernest Hollings*** szenátor terjesztett elő. A CBDTPA messzebbre megy, mint a DMCA, mert megköveteli, hogy minden „digitális médiaeszközt” „szabványos biztonsági megoldásokkal” kell fölszerelni A CBDTPA-t több oldalról is bírálat érte, azonban a szórakoztatóipar szereplői lényegében több jobban emészthető részre darabolták. Az Amerikai Mozgókép Szövetség (MPAA) az elmúlt év során három jogi szabályozás bevezetése érdekében tevékenykedett:

1. „Műsorszórási fejléc” kötelező bevezetése minden digitális műsorszórás vételére alkalmas eszközhöz.
2. Az „analóg kerülőút” kihasználását akadályozó vízjelérzékelő kötelező bevezetése minden olyan eszközön, – digitális átalakítást végez.
3. Egy meghatározatlan műszaki eljárás, amelyet a gép–gép–állománymegosztást végző programokban lenne kötelező használni.

Önmagában e javaslatok egyike sem olyan szigorú, mint a CBDTPA. Az előírásokat fokozatosan is bevezethetik, és elképzelhető, hogy az elektronikai ipar nem indít hadjáratot egyik előírás ellen sem. Lehet, hogy az iparág szereplői megosztottak lesznek, és nem lépnek fel olyan szabályozás ellen, amely közvetlenül nem érinti őket.

Az MPAA máris lobbytevékenységet folytat a „műsorszórási fejléc” bevezetéséért. Ez a javaslat az Egyesült Államokon belüli digitális műsorszórást érinti. De ez az első része egy olyan szabályozási csomagnak, amelyet most kell megakadályoznunk – a filmstúdióknak kevesebb, nem pedig több beleszólásuknak kellene lennie a műszaki megoldásokba.

A jelenlegi amerikai tévéadások az ötvenéves analóg NTSC átviteli szabványt használják. 1996-ban az Egyesült Államok elfogadott egy tervezetet az NTSC leváltására. Az ATSC kiváló minőségű HDTV-képek átvitelére alkalmas. Ennek az átállásnak a terv szerint 2006-ban kell befejeződnie az analóg műsorszórás teljes megszűnésével. Az átmeneti időszakban a tévéállomások analóg és digitális jelet is sugároznak (további csatornákat használva, amelyeket a kormány ad kölcsön). Már kaphatók az ATSC-megfelelő házimozirendszerek és számítógépes vevőkártyák. Az átállás tehát megkezdődött.

Az ATSC nyílt szabvány, ezért a belföldi adások mindig kódolatlanok. A DMCA ez okból kifolyólag nem korlátozza az ATSC-vevőkben megvalósítható képességeket. A hollywoodi stúdiók azt állítják, hogy így az ATSC műsorszórás nem elég biztonságos a hollywoodi filmek közvetítéséhez – ha bárki készíthet vevőkészüléket, olyan módon is használhatják a filmeket, amelyeket a stúdiók meg kívánnak akadályozni.

A stúdiók szerint az ATSC-vevők „kockázata” visszatartja őket attól, hogy ATSC műsorszórásra átadják filmjeiket. Kódolatlan NTSC műsorszórásra fenntartás nélkül átadják őket, de az ATSC adás jobb minősége állítólag jobban csábít a szerzői jogok megsértésére. Az adások kódolása helyett azonban a stúdiók olyan jogi szabályozást javasolnak, amely azt korlátozza, hogy a vevőkészülékek mire használhatják a tévéjelet.

A stúdiók javaslata szerint a tévévevőknek, legyen szó akár az eszközről, akár a programról, az előírásoknak „megfelelőnek” és „ellenállónak” kell lennie a felhasználói módosításokkal szemben. A „megfelelőség” követelménye a megengedhető kimenetet formátumok szűk körére korlátozza, amelyek megvalósítására pedig a DMCA állít fel újabb korlátokat. A nyílt szabványokat nem kötelező alkalmazni. Ez óriási nyomást jelent a szórakoztatóelektronikai cégek számára, hogy jogdíjas megoldásokat alkalmazzanak. A nyílt felületek, amelyekre az egyik stúdió képviselője a „kifutó felületek” kifejezéssel utalt – mivel ezek nem rendelkeznek másolásvédelemmel, jelentős versenyhátrányba kerülnének.

Az „ellenállóság” követelménye meggátolja, hogy a programozók szabad meghajtókat fejlesszenek az ATSC-hez, mint amilyenek már fejlesztés alatt is állnak. Megtiltanák az ATSC szabad programokban történő megvalósításait is, amilyen az FSF-féle GNU Radio Project, ami a rádiójeleket programszinten dolgozza föl. Ha egy programot a felhasználó módosíthat, akkor erre a programra nem bízható annak szabályozása, hogy a felhasználók milyen célra alkalmazzák – ugyanezt az érvet hozza fel az MPAA a szabad DVD-lejátszók ellen is. A műsorszórási fejléc bevezetése éppen azért vezethetne bizonyos szabad programok betiltásához, mert ezeket a programokat a felhasználó módosíthatja.

A Szövetségi Távközlési Bizottság (Fedral Communications Commission, azaz FCC) jelenleg azt fontolgatja, hogy mindenfajta tévé-vevőkészülékben kötelezővé tegye az ilyen előírásoknak megfelelő felszereléseket.

Az EFF küzdelmet folytat e javaslat ellen, és nemrég beadvánnyal fordult az FCC-hez ez ügyben. Részt vettünk a javaslatot kidolgozó Broadcast Protection Discussion Group minden egyes ülésén. „Consensus At Lawyerpoint” nevű weboldalunkkal megpróbáltuk betölteni azt a tájékoztatási űrt, amely a műsorszórási fejléc ügyét körülvette.

Linux Journal 2003. március, 107. szám

Seth David Schoen
Seth hozta létre az EFF személyzeti technológusi beosztást, ez a személy segít más szakembereknek munkájuknak a szabadságjogokra gyakorolt hatásának a megértésében.

Virtuális hálózat egy elosztóval

Sokszor előkerülnek a virtuális hálózatok (VLAN), mégis gyakran nem tudjuk, mire is használhatnánk őket. Cégünknél a minap felmerült egy fura feladat: egyetlen hálózati elosztóval kellett megoldani a különböző munkacsoportok szétválasztását.

A helyzet biztosan ismerős: idővel a vállalaton belül egyre több számítógép gyűlik össze, előbb-utóbb összekapcsolják őket egy hálózatba, egy egyszerű elosztó segítségével. Később kialakulnak a külön munkacsoportok, egy idő után pedig szembekerül a vállalat a hálózati biztonság kérdésével. Üzembe helyeznek ugyan tűzfalat, de mit tehetnek a belső biztonság érdekében? Hogyan tudják szétválasztani a munkacsoportokat, ha nem akarják a már kihúzott kábeleket lecserélni? Másik kérdés lehet, hogy a munkacsoportoknak jó esetben nem szabad látniuk egymást, léteznek viszont olyan kiszolgálók, amelyeket mindegyik gépnek látnia kell. Az ilyen kihívások megoldására több mód lehetséges.

Az első, hogy a különböző munkacsoportokat egy-egy elosztóba (hub) kötjük, külön alhálózatokat hozva létre (*1. ábra*), majd az útválasztást az egyik kiszolgálóra bízzuk. Ehhez a megoldáshoz szükségünk van csoportonként egy elosztóra, megfelelő kábelezésre, valamint egy megbízható kiszolgálóra, melyben legalább annyi hálózati kártya van, ahány alhálózatunk. A megoldással kapcsolatban több gond is felmerülhet: sok kicsi elosztó felügyelete, az elosztókon belül semmiféle felügyeleti lehetőségünk nincs (kivéve, hogy a rosszalkodók gépét kihúzzuk), ugyanígy az alhálózatokon belül az egész hálózati forgalom látszik, valamint az útválasztóként üzemelő gép leállásakor az alhálók közötti kapcsolat megszakad.

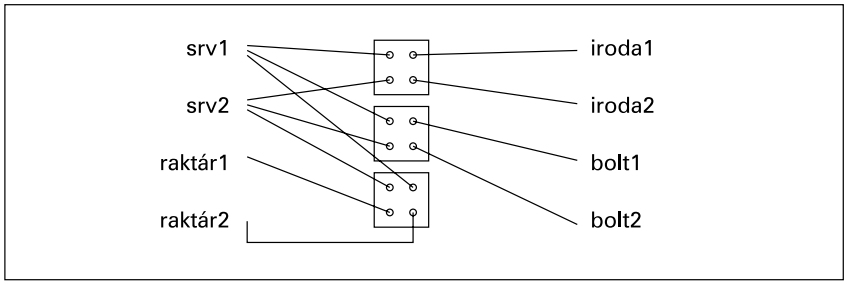
A drága megoldás

A második lehetőségünk, hogy vásárolunk egy céleszközt, egy profi útválasztót, ami külön egységként dolgozik, beállíthatjuk pontosan, hogy ki lát kit, nem kell szenvedni a külön kábelezéssel, egységesen tudjuk felügyelni a hálózatot, sőt, az útválasztó terhelésmentesít is, nem küldi ki feleslegesen mindegyik kapura a csomagokat (*2. ábra*). Emellett azt is meg kell említeni, hogy a céleszközök lényegesen megbízhatóbbak, mint egy PC-alkatrészekből tákolt útválasztó. A hátrányok között csupán egy szerepel, ami viszont néha nagyon komoly gondot okoz, ez pedig az eszköz ára.

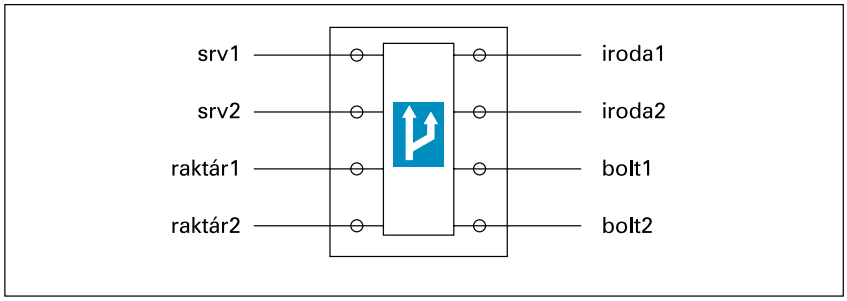
VLAN az útválasztóban

Végül a harmadik megoldás a képzett hálózatok, a VLAN-ok (Virtual LAN) használata. Ilyen „belső hálók” kialakítására több megoldás kínálkozik. A lényeg, hogy valamilyen elmélet szerint csoportokat képezünk a gépekből, és a csoportoknak egyedi azonosítót (VID, VLAN Identifier) adunk. A csoportra bontást megoldhatjuk a hálózati eszközök vagy az ügyfélgépek szintjén. Ahhoz, hogy a hálózati eszközök szintjén oldjuk meg, szükségünk van egy VLAN-csoportosítást támogató útválasztóra vagy kapcsolóra (switch – a kapcsolók lényegesen olcsóbb eszközök, mint az útválasztók, általában kisebb tudással). Ekkor a hálózati eszközben megadjuk, hogy melyik csatlakozójuk melyik képzett csoportba tartozik, a többi pedig az eszköz dolga. Innentől az adott hálózati eszköz úgy működik, mintha sok különálló kisebb egység volna, a képzett alhálózatok pedig nem látják egymást.

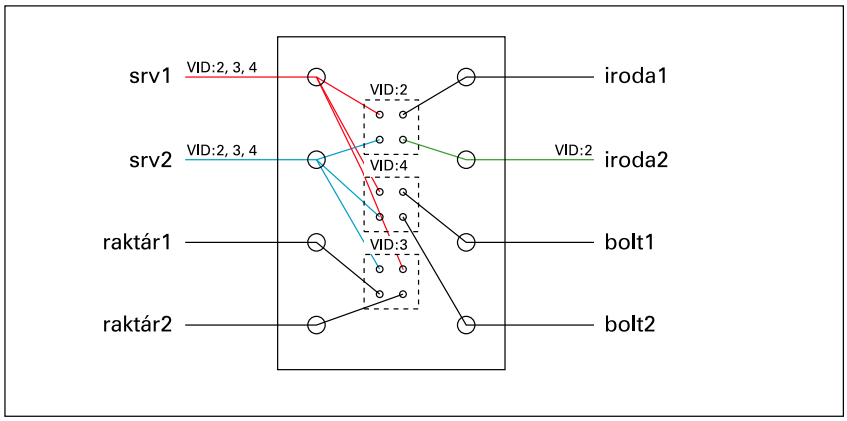
Előnyként lehet felhozni a központosított felügyeletet, az erősebb biztonságot, valamint azt, hogy ma már elérhető áron is lehet kapni VLAN-képes kapcsolókat. Ezzel együtt ennek a megoldásnak is akadnak hátulütői. A legnagyobb baj talán az, hogy a legtöbb elérhető árú kapcsolóval nem tudjuk megadni, hogy egyes kapuk szerepeljenek több csoportban is (mivel egy adott kapuhoz csak egyetlen VID-et adhatunk meg), így ha például egy központi kiszolgálót mindenkinek el kell érnie, akkor a kiszolgálóba annyi hálókártyát kell raknunk, ahány képzett hálóba akarjuk kötni. És ha éppen három gépet akarunk négy képzett hálóból elérni, akkor



1. ábra Három jelelosztó



2. ábra Külön útválasztó



3. ábra Az ügyes megoldás: átfedő VLAN-ok a kapcsolón belül

az tizenkét hálókártyát és a hálózati eszközön tizenkét foglalt kaput jelent.

VLAN-os ügyfélgépek

Megoldhatjuk az alhálók képzését gépszinten is, azonban ehhez további ismeretek szükségesek. A VLAN-ok kitalálásakor az alapötlet az volt, hogy ugyanazon a hálózaton az egymás mellett közlekedő csomagok ne zavarják egymást. Ennek érdekében azok a csomagok, amelyek egy adott képzett hálózathoz tartoznak, módosított szerkezetű IP-fejlécet kapnak. Ugyanazon a hálózaton utaznak tehát az eredeti IP-szabványnak megfelelő csomagok és a VLAN-csomagok is.

Amikor a csomag megérkezik a gépekhez, azok a gépek, amelyek nincsenek felkészítve a VLAN használatára, egyszerűen a „hibás” fejléc miatt eldobják a VLAN-csomagot, míg a képzett hálózatba tartozó gépek visszaalakítják a fejlécet és vígan használják azt. Ennek megoldására régen egy egyszerű módszert használtak, a felismerés és a visszafejtés képességét a hálózati kártyák vezérlőprogramjába építették. Így, ha a húsz gép közül ötöt át szerettünk volna rakni egy külön képzett hálózatba, annyi feladatunk volt csupán, hogy egy másik VID-et adtunk meg az öt gép hálózati kártyájához tartozó vezérlőprogramnak. No igen, rossz esetben pedig vásárolhattunk öt VLAN-t támogató hálózati kártyát, ha éppen egyszerűbb hálókártyák voltak a gépekben.

VLan Linux alatt

Szerencsére kedvenc operációs rendszerünk, a GNU/Linux képes könnyedén kezelni ezeket a kihívásokat, és bármilyen hálózati kártyával egyszerűen részt tudunk venni bármelyik VLAN életében. A Linux-rendszermag képes a legegyszerűbb hálózati csatolón is VID-es csomagokat küldeni. Ehhez a rendszermagba kell fordítanunk a `802.1q` lehetőséget, ezt a ***Networking Options ->802.1Q VLAN Support*** menüpontnál találhatjuk. Miután rendszermagunkat rendben lefordítottuk, próbáljuk is ki ezt a remek kiegészítést!

A VID-vel rendelkező hálózati kapcsolatok az `ethX.n` formát fogják követni, ahhol az `ethX` a használt ethernetkártya (`eth0`, `eth1` stb.) a `.n` pedig a VID-azonosítója.

A beállításhoz használjuk a `vconfig` parancsot, az alábbi példában készítünk egy 2-es VID-vel rendelkező `eth0.2`-es felületet, ami a 10.1.2.0-s alhálózathoz kapcsolódik, beállítjuk hozzá az útválasztó táblát:

```
vconfig add eth0 2;
ifconfig eth0.2 10.1.2.113
netmask 255.255.255.0;
ifconfig eth0.2 up;
route add -net 10.1.2.0
netmask 255.255.255.0 dev
eth0.2
```

Ezek után az `eth0.2`-es csatolón kiküldött csomagok a 10.1.2.0-s alhálózatra fognak menni. A képzett hálózatok között pedig egy linuxos gép vígan elláthatja az útválasztó szerepét. Eljutottunk tehát oda, hogy egyetlen hálózati szakaszon több képzett alhálónk lehet, egy-egy gép pedig részt vehet akár több alháló életében is. És ehhez még nem is kellett módosítani a hálózati felépítést sem! Ennek a kiépítésnek is vannak nehézségei. Itt ismét a legfontosabbat emelem ki először, azt, hogy mindegyik ügyfélgépet egyesével kell beállítani. Fontos még az is, hogyha továbbra is elosztókkal és nem kapcsolókkal vagy útválasztókkal dolgozunk, akkor az összes hálózati forgalom az egyetlen szakaszt terheli, ez pedig komoly korlát lehet.

Kecske és káposzta

Tudom, hogy ennyi lehetőség után már zsong a rendszergazda feje és azon gondolkodik, hogy miért nem tudtak kitalálni valami olyan megoldást, amelyik ötvözi a különféle megoldások hasznát, elsősorban:

- ne kelljen újrakábelezni,
- a képzett hálók ne lássák egymást,
- legyen lehetőség központosított karbantartásra,
- ne kelljen a munkagépeket egyesével piszkálgatni,
- a kiszolgálókba is csak egy hálókártyára legyen szükség,
- a hálózati terhelés legyen elosztva,
- olcsó legyen.

Nos, amikor egyik ismerősömmel beszéltem erről a kérdésről, egy nagyon érdekes megoldást javasolt. A gond az, hogy egy kapcsoló (több okból is kifolyólag) sokszor nem tudja megállapítani, hogy hova kell küldenie egy csomagot. Az volna a legszebb, ha valami ilyesmit mondhatnánk az eszköznek: van hat

munkagépünk (iroda1, iroda2, raktár1, raktár2, bolt1, bolt2) és két kiszolgálónk (srv1, srv2). Tegyük fel, hogy az iroda a 10.10.2.0/24-es hálózat (vagyis a 10.10.2.* című gépekből áll), a raktár a 10.10.3.0/24-es, a bolt a 10.10.4.0/24-es, a kiszolgálók pedig a 10.10.1.1 és 10.10.1.2 címen vannak. Azt szeretnénk, hogy a három részleg (iroda, raktár és bolt) ne lássák egymást, de mindenki lássa a kiszolgálókat. Az egyik megoldás az lehet, hogy a hálózati eszközben bonyolult útvonaltáblákat alakítunk ki, külön engedélyezve a kívánatos irányokat és tiltva a többit. Ehhez az útválasztónak tűzfalként kell üzemelnie.

Egymást átfedő VLAN-ok

Még egy lehetőségünk van, az egymást átfedő VLAN-ok használata (*3. ábra*). Ezzel a megoldással tudjuk kiküszöbölni a már említett tizenkét hálózati kártya esetét, ezért cserébe viszont meg kell oldanunk, hogy kiszolgálóink egy kártyán mindig a megfelelő VID-del közvetítsék az adatokat.
Szerencsére több VID használata könnyedén megoldható Linux alatt a már ismertetett módon. Így semmi másra nincsen szükségünk, csupán a kiszolgálókon az útválasztó tábla beállítására (feltesszük, hogy az `eth0.2` esetén `VID=2` stb.):

```
# route add -net 10.10.2.0
  ➥netmask 255.255.255.0
  ➥dev eth0.2
# route add -net 10.10.3.0
  ➥netmask 255.255.255.0
  ➥dev eth0.3
# route add -net 10.10.4.0
  ➥netmask 255.255.255.0
  ➥dev eth0.4
```

Ezzel elértük, hogy a kiszolgáló az ügyfélgépet mindig a megfelelő VID-vel kezelje. Már csak a másik oldalon kell megoldanunk a feladatot. És mint ahogy már többször is éltem ezzel a szófordulattal a cikk folyamán, itt is két lehetőségünk van. Az első és kevésbé biztonságos, valamint sok ügyfélgép esetén finoman fogalmazva is macerás megoldás, hogy mindegyik ügyfélgépen beállítjuk a kívánt VID-et, ezzel valamelyik képzett hálóba helyezve őt. Ennek hátulütőiről már beszéltünk, mind a hálózati terhelés, mind a felügyelet, mind a biztonság szempontjából meg kell vizsgálni a helyzetet.
Jobb megoldás, ha beszerzünk egy elérhető árú hálózati kapcsolót, mely támogatja az átfedő VLAN-okat. Ennek lényege, hogy az egyes kapukra érkező csomagokat a kapcsoló aszerint kezeli, hogy milyen VID-vel érkezett. Tehát egyetlen kapura érkező forgalom tartalmazhat több képzett hálózathoz tartozó csomagot is. Mi is beszereztünk egy ilyen szerkezetet, egy DLINK gyártmányú 24 kapus kapcsolót.
Ez a hálózati kapcsoló a maga elérhető ára mellett meglepően okosnak bizonyult (ugyanezt az eredményt elérhetnénk más úton is, de például három külön elosztót és egy tűzfalként működő PC-t használva is drágábban jönnénk ki, ráadásul hat-nyolc gép fölött alacsonyabb hálózati teljesítménnyel). Most a mi szempontunkból az a fontos, hogy, az egymást átfedő képzett hálókat is támogatja. Lássuk hogyan!
A kapcsolóban tetszőleges számú képzett hálót kezelhetünk. Ezeket a VLAN-csoportokat egyedi VID-jeik azonosítják. A kapcsoló egyes kapuira érkező csomagok lehetnek címkézett, VID-vel ellátott csomagok, illetve címkézetlenek (eredeti IP-fejlécekkel). Megtehetjük például, hogy az iroda1 géphez vezető kapu címkézett adatokat fogad (mert egy fél évvel ezelőtt az irodai gép hálózati kártyájának megadtuk a `VID=2` beállítást, és lusták vagyunk átállítani), de az iroda2 címkézetlen csomagokat használ (új gép, egy buta hálózati kártyával, ami nem is támogatja a VLAN-okat). A kapcsoló saját maga leszedi a címkét az iroda1 csomagjairól, és címkézetlenül küldi majd ki az iroda2-nek.
Nyilván a kapcsolóban egy adott kapu címkézetlenül csak egy VLAN-ba tartozhat, másként a kapcsoló nem tudná eldönteni, hogy melyik gépeknek küldheti ki az adott gépről érkező csomagokat. Amennyiben viszont címkézett csomagokat kap, azokról nem kell eldöntenie, hogy melyik képzett hálózatba kell küldnie őket, hiszen a címke tartalmazza a megfelelő VID-et.
Ezzel kapcsolatban fontos megjegyezni, hogy a kapcsolón általában egy kapu vagy csak címkézetlen, vagy csak címkézett csomagokat kezel, illetve általában a címkézetlen csomagokat az alapértelmezett VID=1 csoport tagjának tekinti. Ez akkor válik érdekes kérdéssé, ha több kapcsoló vagy útválasztó van jelen a hálózatban. Ilyen esetekre külön protokollokat (például GVRP) biztosítanak a hálózati eszközök.
Eredeti példánk megoldásához nem kell tehát mást tennünk, mint létrehozzuk a szükséges csoportokat, majd hozzárendeljük a kapukat a megfelelő csoportokhoz:

- az srv1 és srv2 gépek címkézett csomagokat forgalmaznak, a `VID=2,3,4` csoportokban,
- az iroda1 címkézve, az iroda2 gép címke nélkül a `VID=2` csoportba tartozik,
- a raktár* és a bolt* gépek pedig címke nélkül a `VID=3`, illetve `VID=4`csoportba tartoznak.

A kiszolgálóknak természetesen látniuk kell egymást is, ehhez vagy kapnak egy külön csoportot, vagy például közvetlen hálózati kapcsolatban állnak egymással (így nem terhelik a munkagépek felé menő sávszélességet az egymás közötti forgalmazással sem).
Ezzel a megoldással tehát nagymértékben fokozzuk a hálózat biztonságát, rendezettségét, lényegesen könnyebb karbantartani az egész hálózatot, és csupán a kiszolgálókon kellett módosítanunk a beállításokat. Ha véletlenül kénytelenek vagyunk egy olyan kiszolgálót is a hálózatba kötni, amelyik nem képes egyetlen hálózati csatolón keresztül több VLAN-t kezelni, akkor még mindig ott a lehetőség, hogy akár több kártyát tegyünk az adott gépbe, akár az egyik értelmes (például linuxos) kiszolgálót átjáróként használjuk.
Szembe kerültünk viszont egy érdekes jelenséggel. Valamilyen, egyelőre a harmadik gamma kvadránsból származó egyedeknek tulajdonítható tevékenységből kifolyólag az egyik kiszolgálónkon, a VLAN-használat bekapcsolásakor a gépeken jelen lévő Virtual LAN beállítások (például az `eth0:1`, `eth0:2` stb. használatához szükséges rész) fellázadtak és a teljes hálózatkezelő rész beadta a kulcsot. Mivel ez a terület egyelőre erősen fejlesztés alatt áll, javasolt a két módszer együttes használatának kerülése.
Első körben ennyit érdemes tudni mindenkinek a vállalati hálózatokban használatos képzett hálók, a VLAN-ok kialakításáról, mely segítségével gyorsan és hatékonyan tudjuk növelni a hálózat belső biztonságát egyetlen közeg esetén is, ahol több csoport és több kiszolgáló is működik. Természetesen a témát messze nem fedtük le, de remélem, elég ötletet mutattunk egy közepes vállalati hálózat megtervezéséhez és felépítéséhez.

Szy György
(Szy.Gyorgy@linuxvilag.hu)
Már hosszú ideje kedvenc szórakozása, hogy minél kevesebb géppel tudja megoldani a szerkesztőség „gépigényét”.

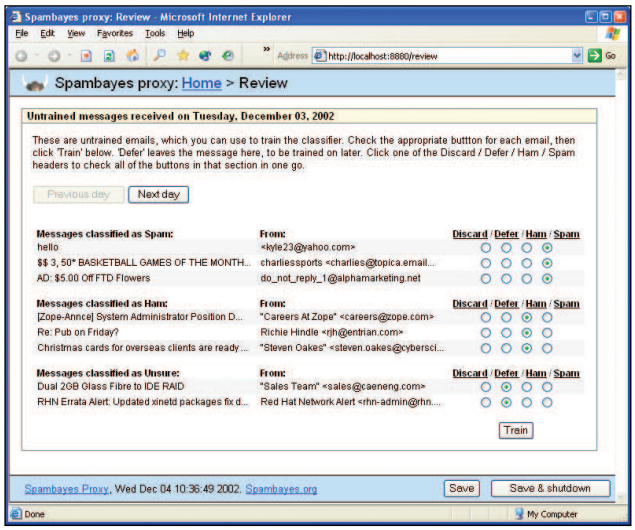
A Spambayes Projekt bemutatása

Egy tanítható rendszer, amely meglévő levelezőrendszerünkkel együttműködve segít elfogni és szűrni a kéretlen leveleket.

A Spambayes Project egy a számos projekt közül, amelyet ***Paul Graham*** „A Plan for Spam” (➲ http://www.paulgraham.com/spam.html) című írása ihletett. Ez a híres cikk arról szól, hogyan lehet a „Bayes-féle elemzés” nevű statisztikai módszerekkel megállapítani egy levélről, hogy kéretlen-e. Aki a Spambayes mögött meghúzódó pontos matematikai eljárás működésére és megvalósítására kíváncsi, olvassa el ***Gary Robinson*** ide vonatkozó cikkét az 24. oldalon.
Dióhéjban: a rendszert ismert kéretlen (spam) és ismert értékes (ham) üzenetek készletével tanítjuk. Az üzeneteket nyelvi egységekre (token) bontjuk, majd minden szóhoz valamilyen értéket rendelünk aszerint, hogy a különböző típusú üzenetekben milyen gyakorisággal fordul elő. Az adatokat adatbázisban tároljuk. Az új üzeneteket egységekre bontjuk, majd a nyelvi egységeket összehasonlítjuk az adatbázisban tároltakkal, s ennek alapján osztályozzuk az üzenetet. Együtt a nyelvi egységek egy összevont pontértéket adnak annak a valószínűségét, hogy a levél kéretlen.
A Spambayes egyik nagy előnye, hogy a rendszer tanításához a saját üzeneteinket használhatjuk fel. Megtanulja az általunk kapott üzenetfajtákat, legyenek azok akár értékes, akár kéretlen levelek.
A Spambayes program az ***X-Spambayes-Classification*** fejléc beszúrásával osztályozza az üzeneteket. A fejléc értéke kéretlen (***spam***), értékes (***ham***) vagy bizonytalan (***unsure***) lehet. A fejléc értéke alapján aztán saját levelezőprogramunkkal elvégezhetjük a szűrést. A kéretlenség mértékét 0-tól (értékes) 1-ig (kéretlen) osztályozzuk. Alapértelmezés szerint a `< 0,2` jelenti az értékes és a `> 0,9` érték jelenti a kéretlen leveleket. Bármely más levél, ami a két érték közé esik, bizonytalan jelölést kap. Ezeket a határértékeket saját magunk is beállíthatjuk; a program beállítására vonatkozó útmutatást az alábbiakban találjuk.

Mitől más a Spambayes?

A Spambayes rendszert három dolog különbözteti meg a többi kéretlenlevél-osztályozótól: próbaalapú tervezési stílusa, nyelvi egységekre bontó algoritmusa (tokenizer) és az osztályozója.
Valamennyien ismerjük a kéretlen levelek nyilvánvaló azonosítási lehetőségeit: üvöltöző címsora van, ami arról értesít bennünket, hogyan Csinálj Pénzt Könnyen!!!, vagy azt sejteti, hogy a nigériai alelnöktől vagy a feleségétől érkezett. Csábító lehetőség kéretlenlevél-osztályozó programunkat ezekre a nyilvánvaló szabályokra alapozni. Például nyilván érdemes megkülönböztetnünk a kis- és nagybetűket, hiszen az INGYEN kifejezés sokkal kéretlen-gyanúsabb, mint az ingyen (még inkább igaz ez angol FREE – free párosra – a ford.). A Spambayes csapat azonban a kezdetektől fogva visszautasította, hogy pusztán megérzésekre alapozza a fejlesztést. A program egyik legkorábbi alkotóeleme egy erős tesztelési keretrendszer, ami képes volt az új ötleteket összehasonlítani a régiekkel. Azokat az ötleteket, amik nem tették jobbá az eredményt, elvetették. Az eredmény gyakran meglepő volt; például a kis- és nagybetű megkülönböztetése nem jelentett számottevő különbséget.
Ez a „bizonyítson vagy pusztuljon” hozzáállás végül csekély felesleges erőfeszítés árán hihetetlenül pontos rendszert eredményezett.
A nyelvi egységre bontó (tokenizer) feladata az üzenetek nyelvi egységekre bontása. Az egyszerű „vágjuk a szóközöknél” rendszerből valami olyasmivé fejlődött, ami az üzenet szerkezetét is ismeri, például képes megjelölni a címsor szavait, hogy ennek következtében a levéltestben szavanként azonosítani lehessen őket. Ismeri a tartalmat is, például



Spambayes proxy webes tanítólap

a beágyazott címeket más módszerrel bontja le, mint az egyszerű szöveget. A bontó összes alkalmazott szabálya szigorú ellenőrzésen ment keresztül, és bizonyítottan növeli a pontosságot. Ide tartozik az is, hogy szántszándékkal elrejtünk néhány nyelvi egységfajtát – például kidobjuk a HTML-cikornyákat, illetve alapértelmezés szerint figyelmen kívül hagyjuk a legtöbb fejlécsort is. Meglepő döntés, de próbaeredmények támasztják alá.
Az osztályozó a Spambayes statisztikai magja, a számdaráló. Paul Graham cikke óta jelentős fejlődésen ment keresztül, úgyszintén komoly próbaalapú fejlesztés eredményeképpen. Gary cikke a „A Bayes-féle levélszűrés matematikai hátteréről”, részletesen foglalkozik az osztályozóval.

Követelmények és telepítés

A Spambayes program a ➲ http://sf.net/projects/spambayes címről tölthető le. Python 2.2 vagy frissebb változatot igényel, ezenkívül a Python ***e-mail*** csomagjának 2.4.3 vagy újabb változata is használja. Amennyiben Python 2.2.2-t vagy frissebb változatát használjuk, ezt a részt az alapcsomag is tartalmazza. Ha mégsem, akkor a ➲ http://mimelib.sf.net címről letölthetjük és telepíthetjük. Kicsomagoljuk a fájlt, `cd`-vel belépünk az

email-2.4.3 könyvtárba, majd kiadjuk a `setup.py install` parancsot. Máris feltelepítettük a Python a saját csomagkönytárába. A hagyományos ***e-mail*** könyvtárat viszont félre kell állítanunk; lépjünk be a ***Python Lib*** könyvtárba, és nevezzük át az ***e-mail*** fájlt ***email_old***-ra.

Hogyan maradjunk naprakészek?

Minthogy a projekt folyamatosan fejlődik, minden bizonnyal számos dolog megváltozik a cikk írásától a magazin újságos standokra kerüléséig eltelt idő alatt. A nagyobb változások összefoglalóját folyamatosan közreadom majd a ➲ http://www.entrian.com/spambayes frissítőlapon.
Miközben ezt a cikket írom, éppen a következő néhány dologgal foglalkozunk: rugalmasabb parancssoros tanítás; beilleszthetőség más levelezőügyfelekbe, például a Muttba; webalapú beállítások; biztonsági lehetőségek a webfelülethez és könnyebb telepítés. Frissítőlapomon mélyrehatóan foglalkozom majd ezekkel a fejlesztésekkel.

Összetevők

A Spambayes program három osztályozót tartalmaz: a Procmail szűrőt, a POP3-proxyt és a Microsoft Outlook 2000 bővítményt (plugint). Ebben a cikkben a Procmail szűrőt és a POP3-proxyt fogom részletezni. A Spambayes részét képezi továbbá a webes felület (amiről később még szó esik) és különféle parancssoros eszközök, próbaeszközök – bővebb útmutatóért olvassuk el a programmal érkezett leírásokat.

Procmail alapú megoldás

Amennyiben Procmail alapú levelezőrendszert használunk, nézzük meg, hogyan is működik a Spambayes procmailes megoldása:

- Minden létező levelünk kap egy új ***X-Spambayes-Trained*** fejlécet. A program ebből tudja majd, hogy melyik üzenetet tanulta már meg.
- A program az összes beérkező levelet átvizsgálja. Azokat az üzeneteket, amelyeket kéretlennek minősít, a kéretlen levelek fiókba helyezi. Minden mást hagyományos módon kézbesít.
- Minden reggel végigmegy a postafiókunkon, és az új levelekkel tanítja magát. Azokat a leveleket is kiválasztja, amelyeket átminősítettünk – azaz amire azt hitte, hogy értékes, pedig valójában kéretlen volt, vagy megfordítva. Törlés előtt legalább két napig vagy még tovább tartsuk meg a ***kéretlen levelek*** fiókba érkezett anyagokat. Az sem árt, ha pár száz levelet megtartunk, hátha újra kell tanítanunk a programot.

A napi tanítási munka végrehajtásához működő `crond`-re lesz szükségünk. Eszményi esetben van egy levelesláda a kéretlen és egy az értékes üzeneteinknek, amelyekkel a kezdeti tanítás elvégezhető.
A Spambayesnek a Procmail rendszer alá történő építéséhez először is telepítsük a programot. Tételezzük fel, hogy a ***$HOME/src/spambayes*** könyvtárba helyeztük. Készítsünk új adatbázist:

```
$HOME/src/spambayes/hammiefilter.py -n
```

Ha élünk a lehetőséggel és úgy döntünk, hogy a már létező leveleink alapján betanítjuk a Spambayest, a következőt gépeljük be:

```
$HOME/src/spambayes/mboxtrain.py
➥-d $HOME/.hammiedb -g $HOME/Mail/inbox
➥-s $HOME/Mail/spam
```

Tetszés szerint további könyvtárneveket is használhatunk, a `-g` kapcsolót használjuk a jó mappákhoz, a `-s` kapcsolót pedig a kéretlen könyvtárakhoz. Ezt követően a következő két képletet kell a ***.procmailrc*** fájl tetejére helyeznünk:

```
:0fw
| $HOME/src/spambayes/hammiefilter.py

:0
* ^X-Spambayes-Classification: spam
$HOME/Maildir/.spam/
```

Az előző képlet Maildir üzenetformátumhoz íródott. Ha nekünk `mbox` (számos rendszeren ez az alapértelmezett) vagy MH kell, a második képlet valahogy így nézne ki:

```
:0:
* ^X-Spambayes-Classification: spam
$HOME/Mail/spam
```

Ha nem vagyunk biztosak benne, hogy melyik formátumot használjuk, kérdezzük meg a rendszergazdát. Ha ön a rendszergazda, ellenőrizze levelezőprogramjának a leírását. A legtöbb korszerű levelezőprogram a Maildir és az `mbox` formátum kezelésére egyaránt képes.
A `crontab -e` parancs segítségével adjuk a következő utasítást a `crontab`-hoz, hogy a Spambayes minden éjjel 2:21-kor tanulhasson az új és az átminősített üzenetekből:

```
21 2 * * * $HOME/src/spambayes/mboxtrain.py -d
➥$HOME/.hammiedb -g $HOME/Mail/inbox-s
➥$HOME/Mail/spam
```

Itt is felsorolhatunk további könyvtárneveket. Ha leveleinket rendszeresen különböző könyvtárakba másoljuk, különösen fontos, hogy ezt megtegyük, máskülönben a Spambayes ezekből a levelekből semmit sem tanul.
A Spambayes ettől kezdve az összes levelünket szűrni fogja, és a levelesládánk alapján tanítja magát. Néha azonban egy-egy üzenet rossz helyre kerül. Az üzenetet egyszerűen csak helyezzük át a helyes könyvtárba, és a Spambayes a következő reggelen tanulni fog a hibájából.
Köszönettel tartozom ***Neale Pickett***-nek az e témakörben nyújtott tájékoztatásért.

A POP3-proxyfelállítása és a webfelület

Ha nem használjuk a Procmailt vagy nem szeretnénk bajlódni vele, esetleg a rendszert nem Unix alapú gépekhez akarjuk használni, használhatjuk a POP3-proxyt. Ez az a közbülső elem, ami a POP3-kiszolgáló és a levelezőprogramunk közt található, és ez fogja letöltés közben az ***X-Spambayes-Classification*** fejlécet a levelekhez csatolni. A POP3-proxyt használhatjuk Fetchmaillel is; egyszerűen csak állítsuk át a Fetchmailt, hogy a POP-proxyval beszélgessen a valódi POP3-kiszolgáló helyett.
A webfelület lehetővé teszi a rendszer előtanítását, az üzenetek osztályozását és a POP3-proxyn keresztül kapott levelek alapján történő tanítást, mindezt böngészőn keresztül.
A programot a ***bayescustomize.ini*** nevű fájl segítségével állíthatjuk be (a frissítő honlapon már olvasható, hogy ehhez is készült webes felület – a fordító). Ez egyébként a Procmail-szűrőre is igaz. Alapesetben szükségtelen az alapértelmezett értékeken változtatnunk, de a POP3-proxyt a POP3-kiszolgálónak megfelelően kell beállítanunk. Az összes lehetséges

beállítás és a hozzájuk rendelt alapértelmezett értékek az ***Options.py*** nevű fájlban kaptak helyet, de ide csak akkor kell benéznünk, ha leküzdhetetlen kíváncsiságot érzünk, vagy kifinomult finomhangolásba kezdtünk. A legkisebb elvárható követelmény az, hogy a ***bayescustomize.ini*** fájlt a következő formában elkészítsük:

```
[pop3proxy]
pop3proxy_servers: pop3.pelda.com
```

A ***pop3.pelda.com*** lesz az a cím, ahonnan a jelenlegi levelezőügyfelünk beállításai alapján begyűjtjük a leveleket. Alapértelmezés szerint a proxy a 110-es kapun fut. Nem unixos gépeken ez megfelelő, Unix alatt azonban nem árt – a következő sort a ***bayescustomize.ini*** fájl `[pop3proxy]` szakaszába beszúrva – megváltoztatni:

```
pop3proxy_ports: 1110
```

Ha több mint egy POP3-kiszolgálóról gyűjtünk leveleket, a `pop3proxy_servers` alatt vesszővel elválasztott címlistát, illetve a nekik megfelelő vesszővel elválasztott kapuszámokat is megadhatjuk a `pop3proxy_ports` alatt. Minden kapu a megfelelő POP3-kiszolgálóhoz alakít ki proxyt. Most már lefuttathatjuk a `pop3proxy.py` parancsot. Néhány állapotüzenetet fog megjeleníteni, körülbelül a következő formában:

```
Listener on port 1110 is
    proxying pop3.example.com:110
User interface url is http://localhost:8880
```

Ami azt jelenti, hogy a proxy készen áll a levelezőügyfelek fogadására a 111-es kapun, a webfelület pedig felkészült, hogy böngészőnket a megadott címre irányítsuk. Ha egy másik gépről akarjuk elérni a webes felületet, a localhost szót helyettesítsük a `pop3proxy.py`-t futtató gép nevével.

Levelek osztályozása a POP3 Proxy segítségével

Be kell állítanunk a levelezőügyfelünket, hogy a leveleket a proxyról, és ne a POP3-kiszolgálónkról gyűjtse be. Ahol eddig POP3-kiszolgálóként a ***pop3.pelda.com*** cím állt a 110-es kapuval, oda most a localhost cím és a 1110-es kapu kerül. Ha a proxyt és a levelezőügyfelet különböző gépeken futtatjuk, akkor a gépnevet és a 1110-es kaput kell megadnunk.
Ettől kezdve leveleink osztályozása mindössze annyiból áll, hogy a ***Levelek letöltése*** gombra kattintunk. A proxy beszúrja a ***X-Spambayes-Classification*** fejlécet az üzenetbe, így levelezőprogramunkban beállíthatunk egy szűrőt, hogy a kéretlenlevél-gyanús leveleket külön mappába helyezze. Természetesen amíg semmilyen tanítást nem végeztünk, összes üzenetünk bizonytalan kategóriájú lesz.

Tanítás webfelületen keresztül

A kezdeti tanítás nem kötelező előfeltétel, de az elejétől fogva jobb eredményt kapunk, ha végrehajtjuk. Webfelületes tanításnál az üzenetek vagy az ***mbox fájl feltöltése*** űrlapot használhatjuk egyedi üzenetek vagy Unix mbox fájlok elemzésére. Amint működik a rendszerünk, a webfelület segítségével a POP3-proxy által már látott üzenetekkel taníthatjuk a rendszert. A ***Review messages*** (üzenetek vizsgálata) lapon megtaláljuk osztályozott üzeneteinket aszerint csoportosítva, hogy a program kéretlen vagy értékes levélosztályba sorolta-e őket, esetleg bizonytalan volt-e. Itt kijavíthatjuk a hibákat, ha beállítjuk a jelölőnégyzetet, majd a ***Train*** (tanítás) gombra kattintunk. Néhány nap múlva (attól függően, hány levelet szoktunk kapni) csak nagyon kevés hiba fog keletkezni.

Tanítási tippek

- Ne tanítsunk régi levelet. Leveleink jellemzői idővel változhatnak – néha alig, néha nagymértékben –, ezért az a legjobb, ha csak friss leveleket használunk.
- Figyeljünk, amikor tanítunk. Ha hibásan értékesnek tüntetünk fel egy kéretlen levelet, vagy megfordítva, komolyan hátráltatjuk az osztályozót.
- Próbáljunk meg ugyanannyi kéretlen és értékes levelet kiválasztani. Ez ugyan nem létfontosságú, de jobb eredményt kapunk, ha fenntartjuk az egyensúlyt.

Lehetséges fejlesztési irányok

A Spambayes program folyamatosan fejlődik. Egy sereg ember vesz részt benne, és rengeteg ötletünk van, milyen módon fejlődjünk tovább. Egy kis ízelítő a lehetséges fejlődési irányból:

- A nyelvi egységre bontó és az osztályozó fejlesztése az új kutatások szerinti pontosabb kéretlenlevél-osztályozás érdekében.
- Intelligens önműködő tanítás: ha a rendszer fut, lehetővé válna, hogy saját magát tanítva váljon naprakésszé, és a felhasználóknak csak a fura hibákat kell kijavítaniuk. A Procmail rendszerrel már meg is valósítottunk valami ilyesmit, de keressük a további lehetőségeket, amelyekkel még önműködőbb és felületfüggetlenebb megoldást készíthetünk.
- SMTP-proxy: a rendszer tanítása bármilyen levelezőügyfélből bármilyen felületen, ahol az üzeneteket különleges értékes vagy kéretlen levélcímekre küldhetjük el. Ez egyszerű módja lenne az osztályozási hibák javításának, és jól összeegyeztethető az intelligens öntanuló módszerekkel.
- Adatbázisméret-csökkentés: minél többet tanítjuk a rendszert, annál nagyobbá válik az adatbázis. Keressük a módját, hogyan lehetne kis adatbázisméretet tartani.
- Egybeépítési lehetőség a levélszemét-jelentő eszközökkel: jó lenne, ha a webfelület és a levelező bővítmény (plugin), hogy jelentéseket küldjünk az olyan rendszereknek, mint a Vipul Razorja vagy a Pyzor.
- További levelezőügyfél-beépítések: Outlook-bővítményünk már van, de további levelezőügyfeleket is szeretnénk támogatni. A POP3-proxy és a webcsatolófelület minden levelezőügyféllel jól működik, de egy levelezőnkbe épített ***Delete as Spam*** gomb sokkal kényelmesebb, mint átváltogatni a böngészőnkre.
- Jobb leírás: szeretnénk kiadni egy dokumentációt arról, hogyan kell a Spambayest a népszerű felületeken és a levelezőügyfeleken beállítani.

Mire ez a cikk a nyomdába kerül, a fent említettek közül sok talán meg is valósul; a részletek a ➲ http://www.entrian.com/spambayes a frissítési oldalon találhatók.

Linux Journal 2003. március, 107. szám

Richie Hindle
Hivatásos programmérnök. Teljes állásban üzletiintelligencia-programokat ír, szabadidejében pedig a Spambayes rendszeren és a ➲ http://www.entrian.com címen fellelhető, saját Python-projektjein dolgozik.

A Bayes-féle levélszűrés matematikai hátteréről

A legtöbb ember naponta jelentős időt tölt a hasznos levelek és a kéretlen reklám (spam) szétválogatásával. A levélszemétszűrő programok ezen a gondon segíthetnek.

Ebben a cikkben a kéretlen levelek szűrésének alapjául szolgáló matematikai eljárások közül csak egyet vizsgálunk meg – a levél tartalmát meghatározó jelek gyűjteményéből létrehozunk egy „kéretlenség”-mutatót. Az itt leírt megoldás a legjobb nyílt forrású hagyományoknak megfelelően valódi közös munka volt. ***Paul Graham***, több Lispről szóló könyv szerzője, „A Plan for Spam” című hálózati cikkében javasolt egy kéretlenlevél-szűrési megoldást. Ezt a megközelítést vettem át a szavakhoz rendelt valószínűségek meghatározásához, kicsit módosítottam, majd a nem túl gyakran használt szavak kezelésére bevezettem a Bayes-féle számítást. Később felvetettem, hogy próbáljuk meg a chinégyzet-eloszlást és az egyes szavak valószínűségét egyetlen kombinált valószínűségbe (valójában valószínűségpárba – lásd később) összerakva felhasználni, és így jellemezzük a levelet. Végül ***Tim Peters*** a Spambayes Projektből bemutatott egy olyan módszert, amellyel különösen jól használható kéretlenségmutatót készíthetünk kombinált valószínűségek segítségével. A munkát folyamatos próbák sora kísérte a Spambayeshez fejlesztő Tim Peters Pythonban írt és a Bogofilter Projekten dolgozó ***Greg Louis*** C nyelvű megvalósításain. A kipróbálást a fent nevezett projektekben résztvevő emberek sokasága végezte.

Szóvalószínűségek előállítása

Feltételezzük, hogy a levelekben találunk a tanuláshoz szükséges testet, és van olyan programunk, ami képes az egyes leveleket alkotóelemeire (szavaira) bontani. Ezenkívül feltételezzük még azt is, hogy minden tanítólevelet korábban már kézi módszerrel osztályoztunk, a „ham” (***értékes***, vagyis olvasandó levelek) és a „spam” (***kéretlen***, azaz szemét, tehát olyan levél, amit nem akarunk elolvasni) csoportokba sorolva be őket. Ezeket az adatokat és programokat felhasználva fogjuk a rendszerünket megtanítani, minden szóhoz a kéretlenségét jelző valószínűséget hozva létre.
A levéltestben előforduló valamennyi szóra kiszámítjuk a következő értékeket:

- `b(w)` = (a `w` szót tartalmazó kéretlen levelek száma) / (az összes kéretlen levél darabszáma).
- `g(w)` = (a `w` szót tartalmazó értékes levelek száma) / (az összes értékes levél darabszáma).
- `p(w)` = `b(w) / (b(w) + g(w))`

A `p(w)` értéket megközelítőleg tekinthetjük annak a valószínűségnek, hogy a *w* szót tartalmazó találomra kiválasztott levél kéretlen lesz. A kéretlenlevél-szűrő programok a levélben található minden egyes szóra kiszámolhatják a `p(w)` értéket, majd a levél kéretlenségének megállapításához felhasználhatják a további számításaikban.
Akad azonban egy bökkenő: a valós világban egy adott személy postafiókjában tíz, de akár kilencven százaléknyi kéretlen levél is lehet. Nyilvánvaló módon az adott személy esetében ennek hatása lesz annak valószínűségére, hogy a szót tartalmazó levél ***kéretlen*** vagy ***értékes***-e. Ezt a hatást a fenti számításban figyelmen kívül hagytuk. Ehelyett számításunkban tulajdonképpen egy olyan világban közelítettük a *w* szót tartalmazó, véletlenül kiválasztott levél kéretlenségének valószínűségét, ahol éppen a levelek fele kéretlen és fele értékes. Az ilyen megközelítés előnye azon alapul, hogy nem szeretnénk, ha a leveleket könnyebben vagy nehezebben sorolnánk a ***kéretlen*** osztályba csak azért, mert kapott leveleinkben a kéretlen és értékes levelek aránya nem egyenlő. Ehelyett inkább azt szeretnénk, hogy a leveleket tisztán a saját szerkezetük alapján osztályozzuk. Ez a feltételezés végül is jól bevált a gyakorlatban.

Ritka szavak kezelése

A korábban kiszámolt valószínűségek hibásak lesznek, ha a szavak nagyon ritkák. Például amennyiben a szó pontosan egy levélben szerepel és az a levél kéretlen, a számított `p(w)` pontosan 1,0 lesz. Nyilvánvaló, hogy nem száz százalék annak az esélye, hogy a jövőben kapott, ezt a szót tartalmazó valamennyi levél kéretlen lesz; egyszerűen csak nincs elég adatunk a valós valószínűség meghatározásához.
A Bayes-féle statisztika hatékony eszköz a kezünkben az ilyen esetek kezelésekor. A statisztikák e csoportjának alapja, hogy érdeklődésünk tárgya egy bizonyos esemény bekövetkezésébe vetett hitünk mértéke – ez az esemény Bayes-féle valószínűsége.
Ha pontosan egy levél tartalmazza az adott szót, a következő ilyen szót tartalmazó levélről nem gondoljuk százszázalékos biztonsággal, hogy kéretlen. Ez azért van így, mert egyéb háttéradattal is rendelkezünk, ami segít bennünket. Tapasztalatból tudjuk, hogy tulajdonképpen bármilyen szó előfordulhat mind a kéretlen, mind az értékes levélkörnyezetben is, illetve hogy egy maréknyi minta még nem elég ahhoz, hogy tökéletesen biztosak legyünk egy esemény valószínűségében. A Bayes-féle megközelítés lehetővé teszi, hogy általános háttértudásunkat bevigyük a szóhoz gyűjtött adataink mellé, így mindkét megközelítés az őt megillető helyre kerül. Ezzel a módszerrel azt a helyes értéket állapíthatjuk meg, ami azt a hitünket tükrözi, miszerint amikor a legközelebb látjuk az adott szót, akkor azt egy kéretlen levélben találjuk. Ezt a „reménybeli mértéket” jellemző `f(w)` értéket a következőképpen kapjuk meg:

$$f(w) = \frac{(s * x) + (n * p(w))}{s + n}$$

- Az `s` jelzi, hogy mekkora súlyt szeretnénk adni a háttéradatainknak.
- Az `x` az általános háttéradatainkra alapozott feltételezett valószínűség, ami szerint egy szó, amellyel kapcsolatban még semmiféle tapasztalatunk nincs, kéretlen levélben található.
- Az `n` jelzi a *w* szót tartalmazó levelek számát.

Ezzel a módszerrel egyszerűen használhatjuk az `x` értéket a háttéradatból kikövetkeztetett valószínűség kezelésére, valamint az `s` értéket az adat súlyának a meghatározáshoz. A gyakorlatban az `s` és `x` értékeket a teljesítmény növelése érdekében végzett próbák alapján állapítjuk meg. Kezdeti `s` és `x` értéknek

elég jól megfelel az 1 és 0,5 érték. Számításainkban a `p(w)` függvény helyett az `f(w)` változatot használjuk, hiszen ésszerű valószínűségi értékekkel dolgozunk, és nem a valóságtól elrugaszkodott különleges mennyiségekkel, amelyekkel gyakran találkozhatunk, ha nem áll rendelkezésünkre megfelelő mennyiségű adat. A képlet egyszerű módszert ad a kezünkbe, amivel azt az esetet is kezelhetjük, amikor egyáltalán nem áll rendelkezésünkre adat; ebben az esetben az `f(w)` értéke pontosan a háttéradatok alapján feltételezett valószínűség lesz.
Akiket érdekel a fenti képlet levezetése, folytassák az olvasást; a többiek nyugodtan továbbugorhatnak a következő részhez.
A fenti képlet azon a feltételezésen alapul, hogy a ***w*** szót tartalmazó levelek ***kéretlen***, illetve ***értékes*** osztályozása bétaeloszlással jellemezhető, binomiálisan véletlen változó. Az utólagos elvárást (posterior expectation) a megfigyelt adatok beépítése után számoljuk ki. Olyan próbát akarunk végrehajtani, ami megfelel annak a kísérletnek, amikor többször feldobunk egy pénzérmét, hogy lássuk, ugyanarra az oldalára fog-e esni vagy sem. A vizsgálatok száma ***n***. Ha pénzérmét dobálnánk, minden dobás egy vizsgálat lenne, és a fejek megjelenését számlálnánk. Esetünkben azonban a vizsgálat tárgya a tanító levéltestek közül a következő „porn” szót tartalmazó levél lesz, és azt vizsgáljuk, hogy a szót tartalmazó levél kéretlen-e. Ha az, a kísérletet sikeresnek tekintjük. Egyértelműen binominális kísérletről van szó: két érték létezik, igen vagy nem. A kettő független: az a tény, hogy az adott levél tartalmazza a „porn” szót, nincs összefüggésben azzal, hogy a következő tartalmazza-e vagy sem. Tehát van egy biniominális kísérletünk, ahol az elsőhöz bétaeloszlást tételezünk fel, ennek megfelelően az n + 1-dik kísérlet sikerének várható értékét a második egyenlet alapján határozhatjuk meg.

$$E = \frac{u + q}{u + v + n}$$

- `q` a sikerek száma.
- `n` a kísérletek száma.
- `u` és `v` a bétaeloszlás értékei.

Megmutatjuk, hogy az első egyenlet és a második egyenlet egyenértékű (ekvivalens). Végezzük el a következő behelyettesítéseket:
s = u + v
s × x = u
Majd `q` helyére helyettesítsük az n × p(w) képletet. Korábban már láttuk, hogy a `p(w)` annak a becsült valószínűsége, hogy egy véletlenszerűen választott, ***w*** szót tartalmazó levél kéretlen-e egy olyan képzeletbeli világban, ahol éppen ugyanannyi kéretlen és értékes létezik. Így az n × p(w) az ebben a világban található ***w*** szót tartalmazó kéretlen levelek számát közelíti. Ez a sikereink számának közelítő értéke, így egyenértékű q-val. Ezáltal beláttuk az első és második egyenlet egyenértékűségét.
A kipróbálás során, amikor valamennyi korábban `p(w)`-t tartalmazó képletben `f(w)`-vel helyettesítettük a `p(w)` függvényt, egységesen megbízhatóbb ***kéretlen***, illetve ***értékes*** osztályozást kaptunk.

Valószínűségek összevonása

Immár képesek vagyunk kiszámítani az új levélben található valamennyi szóhoz tartozó `f(w)` valószínűséget. Ez a valószínűség tükrözi a háttéradat és a mintalevéltest adataink alapján meghatározott gyanúnk mértékét, miszerint a véletlenszerűen kiválasztott ***w*** szót tartalmazó levél kéretlen lesz.
Minden levelet tehát egy valószínűségkészlet jellemez. Ezeket az egyedi valószínűségeket szeretnénk egyetlen általános mutatóba összevonni, ami a levél kéretlenségét vagy értékességét fogja jellemezni.
A metaanalízis néven ismert statisztikai területen talán a legelterjedtebb valószínűség összevonási módszer ***R. A. Fisher***-től származik. Jelöljük valószínűségkészletünk elemeit a $p_1, p_2, ..., p_n$ betűkkel. Először is számítsuk ki a $-2\ln p_1 \times p_2 \times ... \times p_n$ értéket. Ezután az eredményt tekintsük $2n$ szabadságfokú chinégyzet-eloszlásnak, majd a chinégyzet-tábla segítségével számítsuk ki azt a valószínűséget, ami ugyanolyan szélsőséges vagy szélsőségesebb értéket adna eredményül, mint a kiszámított érték. Ez az „összevont” valószínűség értelmesen összegzi az összes egyedi valószínűséget.
A valószínűségek kezdeti készletét a nullhipotézis figyelembevételével alkalmazzuk. Például, ha nullhipotézisként azt feltételezzük, hogy az érme elfogulatlan, majd tízszer feldobva egymás után tíz fejet kapunk, az eredményül kapott valószínűség $(1/2)^{10} = 1/1024$ lenne. A nullhipotézisnek megfelelően nagyon valószínűtlen, hogy ez bekövetkezzen, de természetesen ha az érme elfogult volna, nem lenne szükségszerűen ilyen valószínűtlen a szélsőséges esemény. Ezért aztán elvethetjük a nullhipotézist, és helyette nyugodtan elfogadhatunk egy másik hipotézist, ami szerint az érme elfogult.
Ugyanezeket a számításokat felhasználhatjuk az `f(w)` értékek összevonására is. Az `f(w)-k` nem valódi fizikai valószínűségek. Inkább úgy gondolhatunk rájuk, mint a valószínűségek legjobb tippjeire. A Fisher-számítás hagyományos metaanalitikus felhasználásánál szintén nincs szükség arra, hogy a valós valószínűségekkel dolgozzunk. Ehelyett feltételezzük, hogy a nullhipotézis részei. Legyen nullhipotézisünk a következő: „az `f(w)` értékek pontosak, és a vizsgált levél egymástól független véletlen szavak gyűjteménye, így az `f(w)`-k egyenletes eloszlásúak.” Most tegyük fel, hogy az egyik szavunk a „Python”, aminek az `f(w)` értéke 0,01. Úgy gondoljuk, hogy csak az esetek egy százalékában fordul elő kéretlen levelekben. Majd ezt kiterjesztve feltételezzük, hogy valószínűtlen esemény történt; olyan, aminek az esélye mindössze 0,01. Hasonlóképpen a levél minden egyes szavához hozzárendelhetünk egy valószínűséget. Az olyan nagyon kéretlengyanús szavak, mint a „porn”, 0,99 vagy még nagyobb valószínűséggel bírnak.
Ezek után a Fisher-számítás segítségével kiszámítjuk a szavak készletéhez tartozó valószínűséget. Ha a levél értékes, igen valószínű, hogy számos nagyon alacsony valószínűségű, és viszonylag kevés magas valószínűségű szót fog tartalmazni, eredményképpen a Fisher-számítás igen alacsony összesített valószínűséget fog adni. Így lehetővé válik, hogy elvessük a nullhipotézist, és egy másik hipotézist tételezünk fel, miszerint a levél valójában értékes.
Számítsuk ki ezt az összevont H valószínűséget:

$$H = C^{-1}(-2 \ln \prod_{w} f(w), 2n)$$

Ebben a `C-1()` a chinégyzet-függvény inverze, aminek a felhasználásával a chinégyzet-eloszlású véletlen változóból levezetjük a `p` értéket.
Természetesen mi már előre tudtuk, hogy a nullhipotézis hamis. A valóságban egyetlen levél sem áll véletlen szavakból, amik mind függetlenül indulnak az értékes vagy a kéretlen címért; a levél általában számos szót tartalmaz az egyik vagy a másik típusból. Természetesen a szavak sem függetlenek. A „sex” szót tartalmazó levelek sokkal nagyobb eséllyel tartalmazzák a „porn” szót is, és a Pythonban programozó barátunk nevét

Az inverz chinégyzet-függvény megvalósítása Python nyelven

```
import math

def chi2P(chi, df):
    """VisszatØrØsi ØrtØk val sz nßsØg(chinØgyzet >= chi,
      df szabadsÆgi fokkal). df pÆrosnak kell lennie."""
    assert df & 1 == 0
    # XXX ha chi nagyon nagy, az exp(-m)
    # 0-ra s llyed.
    m = chi / 2.0
    sum = term = math.exp(-m)
    for i in range(1, df//2):
        term *= m / i
        sum += term
    # Kis chi Øs nagy df esetØn, a felhalmozott
    # kerek tØsi hiba, plusz a
    # az exp() f ggvØny hibÆja miatt elıfordulhat,
    # hogy nØhÆny ULP-vel 1,0 f lØ nı ez az ØrtØk.
    # PØldÆul a chi2P(100, 300) ØrtØke az Øn gØpemen
    # sum == 1,0 + 2,0**-52 ezen a ponton.
    # Az 1,0-t akÆr csak egyetlen apr  bittel is meghalad
    # ØrtØket visszaadni helytelen megoldÆs.
    return min(sum, 1.0)
```

tartalmazó levelek is nagyobb eséllyel tartalmazzák a „Python” szót. Az `f(w)`-k sem egyenletes eloszlásúak. A kéretlenlevél-szűrés esetében azonban a valóságtól való ilyen mértékű eltérés még megengedhető. Hatásukra a valószínűségek nem-véletlen tendenciát mutatnak, ami egy adott levélben lehet nagy vagy kicsi, erős statisztikai alapot adva nekünk, hogy az egyik vagy másik választható feltevés kedvéért nullhipotézisünket elvessük, azaz eldöntsük, hogy a levél értékes-e vagy kéretlen.
A nullhipotézis tehát olyan, mint a szalmabábu: csak azért állítjuk fel, hogy aztán az egyik vagy a másik irányba leverhessük.
Érdemes megemlíteni egy kulcsfontosságú különbséget a tárgyalt megközelítés és más összevonó megoldások közt, amik a változók függetlenségét tételezik fel. Ez a különbség megjelenik például a „Bayes-féle láncszabály” vagy a „Naive Bayes-féle osztályozása” esetében is. Ezeket a megközelítéseket szemtől szemben versenyeztettük az itt leírt megközelítéssel nagyszámú, emberek által osztályozott levelek felhasználásával, és közel sem teljesítettek ilyen jól (pontosabban nem egyeztek meg ennyire az emberi ítélettel). A két másik megközelítés számításai műszakilag hibásak, mivel olyan adatpontok függetlenségét igénylik, amik nincsenek jelen. Ezt a hibát a Fisher-módszer alkalmazásával elkerülhetjük, mivel a számítások helyessége nem függ az adatok függetlenségétől. A Fisher-számítást úgy építettük fel, hogy elvethetjük azt a nullhipotézist, ami az adatok függetlenségét tartalmazza, két másik, minket érdeklő vagylagos feltevés kedvéért. Ez a két vagylagos feltevés már tartalmazza a nem-függetlenséget, például a „sex” és a „porn” közötti korrelációt a végletes összesített valószínűségeket létrehozó jelenség részeként. Ezek az összesített valószínűségek csak akkor tekinthetők valószínűségeknek, ha a nullhipotézis – amiről tudjuk, hogy majdnem biztosan hamis – igaz. A valós világban a Fisher összesített valószínűségek tulajdonképpen nem is valószínűségek, hanem inkább elvont jellemzői annak, mennyire (vagy milyen kevéssé) voltunk hajlandóak elfogadni a nullhipotézist. Ha a nullhipotézisünk hamis, nem várunk igazi valószínűségeket, így az, hogy valóban nem valódiak, nem okoznak számítási hibákat. A Naive Bayes-féle osztályozó közismerten kevésbé érzékeny a függetlenség hiányából adódó torzulásokra, de nem képes tökéletesen elkerülni őket.
A mai napig nem tudjuk, hogy a Fisher-féle kéretlen/értékes osztályozáspróbák kimagasló teljesítménye vajon az adatokban fellelhető függetlenség hiányával szembeni különböző érzékenységnek köszönhető-e, de ezt is figyelembe kell vennünk, mint egy lehetséges tényezőt. Az egyes `f(w)` értékek csak becslései a valós valószínűségeknek (azaz amikor nagyon kevés adatunk van egy szóról, az `f(w)` által kapott kéretlenségi valószínűségbecslésünk nem feltétlenül tükrözi a valós valószínűséget). De ha belegondolunk, hogyan állítottuk elő az `f(w)`-t, láthatjuk, hogy ez a bizonytalanság aszimptotikusan csökken, ahogy az `f(w)` közelít a 0-hoz vagy az 1-hez, mivel ezeket a szélsőséges értékeket csak akkor érhetjük el, ha igen gyakran fordulnak elő a tanítóhalmazban, ráadásul majdnem mindig kéretlen levélben, vagy majdnem mindig értékes levélben találjuk meg őket. Továbbá látható, hogy a 0-hoz közeli számoknak van messze a legnagyobb hatásuk a számításokra. Ennek megértéséhez vizsgáljuk meg a következő szorzatot: 0,01 × 0,5, ha az első kifejezés 0,001-re változott, szemben az 0,51 ×0,5 szorzattal, amikor az első kifejezés 0,501-re változott, és emlékezzünk, hogy a Fisher-módszer a valószínűségek szorzásán alapul.
Nullhipotézisünk sérül azáltal, hogy az `f(w)` `-k` nem teljesen megbízhatóak, de a módszer miatt ez a rész elenyészően kicsiny lesz a minket érdeklő értékgyanút bizonyító keresett szavakhoz, vagyis az olyan szavakhoz képest, amelyeknek az `f(w)` értéke közel 0.
A figyelmes olvasó ezen a ponton elcsodálkozhat: „rendben, most már értem, úgy tűnik, ezek a Fisher-számítások jól kezelik a közel nulla valószínűségű értékgyanús szavakat, de mi lesz a közel 1 valószínűségű kéretlengyanús szavakkal?” Jó kérdés! A választ alább találjuk, ami egyben le is zárja vizsgálódásainkat.

Az értékes és kéretlen levelek gyanújának jelölése

A fent leírt számítások érzékenyek az értékes levelek gyanújára, főként ha olyan szavak formájában fordulnak elő, amelyek sokkal több értékes levélben fordulnak elő, mint kéretlenben. Ennek az az oka, hogy a 0 közeli valószínűségek hatása sokkal erősebb a végső valószínűség értékére, ami tulajdonképpen a Fisher-számítás lényege. Sőt létezik egy 1971-es tétel, ami kimondja, a Fisher-módszer bizonyos körülmények közt, a valószínűségek szorzata mögötti folyamatokat kutató módszerek közül a leghatékonyabb (lásd a ***Kapcsolódó címeket***). Ugyanakkor az erősen kéretlen gyanús szavak `f(w)` értéke közel 1, így sokkal kevesebb hatással lesznek a számításra. Azt is gondolhatnánk, hogy ez jó dolog. Végső soron, igen sok embernek egy jó levél téves kéretlenné nyilvánítása sokkal rosszabbnak tűnik, mint egy rossz levél téves értékessé osztályozása,

hiszen nagy baj nem származik abból, ha egyetlen kéretlen levél valahogy átcsúszik, de komoly gond keletkezhet abból, ha egyetlen jó levelet is rosszul osztályozunk, és így a fogadó figyelmen kívül hagyja azt. Ennek megfelelően jó ötletnek tűnik, ha érzékenyek vagyunk az értékes levél gyanúját mutatók és kevésbé érzékenyek a kéretlen levelet mutatók irányába.

Ugyanakkor léteznek olyan módszerek, amelyekkel kiküszöbölhetjük a nehézséget, s a valós kipróbálás során nem tapasztaltunk növekedő tendenciát a hibásan kéretlené nyilvánított levelek közt, viszont lényegesen csökkent a hibásan értékesnek nyilvánított kéretlenek aránya. A következőkben bemutatjuk a mostanában végzett próbák során leghatékonyabbnak talált módszert.

Először is „megfordítjuk” az összes valószínűséget, kivonva őket 1-ből (azaz, minden egyes szóra kiszámítjuk az `1 f(w)` értéket). Minthogy az `f(w)` annak a valószínűségét szemlélteti, hogy a levélkészletből véletlenszerűen választott, `w` szót tartalmazó levél kéretlen, az `1 – f(w)` annak a valószínűségét jellemzi, hogy ez a véletlenül választott levél értékes lesz. Most végezzük el a korábban is használt Fisher-számítást, de az `f(w)` értékek helyett most az `(1 – f(w))`-ket használjuk. Eredményül majdnem 0 összesített valószínűségeket kapunk, a nullhipotézis elutasítása esetén, amikor igen sok kéretlen gyanús szó van jelen. Nevezzük ezt az összesített valószínűséget `S`-nek. Számítsuk ki a következő értéket:

$$I = \frac{1 + H - S}{2}$$

Az `I` értéke közel 1, ha a bizonyítékok túlsúlya azt mutatja, hogy a levél kéretlen és közel 0, amikor a bizonyítékok arra engednek következtetni, hogy értékes levélről van szó. Az értéknek akad néhány érdekes jellemzője.

Tételezzük fel, hogy a levél számos kéretlen gyanús szót tartalmaz, ugyanakkor találunk benne számos értékes levélre jellemző szót is. Mivel a Fisher-módszer a 0 melletti értékekre érzékeny és jóval kevésbé veszi figyelembe az 1 közeli értékeket, az eredmény valószínűleg az lesz, hogy az `S` és `H` értékek egyaránt 0 közeliek lesznek. Például az `S` értéke 0,00001 nagyságrendű a `H` értéke pedig 0,000000001 nagyságrendű. Valójában az ilyen típusú eredmények egyáltalán nem olyan ritkák, mint ahogyan azt a valós levelek esetében gondolnánk. Jó példa erre, amikor egy barát továbbküld egy kéretlen levelet egy másik barátjának egy kéretlen levelekről szóló beszélgetés részeként. Ilyen esetben mindkét végkövetkeztetést komoly bizonyítékok támasztják alá.

Sok megközelítésben, például a Bayes-féle láncokon alapuló módszerek esetében az a tény, hogy a példánkban több kéretlenlevél-szerű szót találunk, mint értékeslevél-szerűt, arra készteti az osztályozót, hogy egyértelműen kijelentse: a levél kéretlen. Csakhogy ez egyáltalán nem olyan biztos; példának okáért a továbbított levélről szóló példánk nem kéretlen levél volt.

Így aztán hasznos tulajdonság, hogy az `I` ilyen esetekben 0,5 körüli, illetve ugyanígy 0,5 körüli lesz akkor is, ha egyik irányt sem erősíti meg semmilyen különösebb bizonyíték. Amikor mindkét következtetést erős bizonyítékok támasztják alá, az `I` értéke az óvatos megközelítést tükrözi. A valós világban végzett próbák során ezeknek a középértékű leveleknek az emberi vizsgálata azt az álláspontot támasztotta alá, miszerint ezeket a leveleket valóban valahová középre kell besorolni, és nem szabad a legtöbb osztályozó fekete-fehér megközelítését alkalmazni rájuk.

A Spambayes Projekt, amelyet ***Richie Hindle*** cikke mutat be a 21. oldalon, ki is használja ezt, bizonytalannak jelölve a 0,5 érték közelébe eső leveleket. Ezáltal lehetőség nyílik rá, hogy a levél címzettjének felkeltsük a figyelmét a bizonytalanul osztályozott levelek iránt. Így csökken annak az esélye, hogy egy jó levelet a hibás osztályozás miatt véletlenül kihagyunk.

Jövőkép

Jelenleg a módszert alkalmazó program nyelvi egységenkénti (token) egy szó alapú megközelítést alkalmaz. Más megközelítés is létezhet, például felépíthetünk egy kereső-táblát a frázisokra. Az itt leírt matematikai eljárás várhatóan ebben az összefüggésben is hasonlóan jól teljesít, és joggal hihetjük, hogy a frázisalapú rendszer további teljesítményelőnyökkel jár majd, bár néhányan vitatják az ötlet helyességét. További Linuxvilág-cikkek is várhatók ebben a témakörben, amennyiben e területen továbblépünk. Ilyen frázisalapú rendszerre példa a CRM114 (lásd a ***Kapcsolódó címeket***), ami igen jól teljesít, bár e cikk írásakor még közvetlenül nem próbálták ki, más módszerekkel ellentétben, azonos levéltesteken. (Írásunk keletkezésekor a CRM114 a Bayes-féle láncszabályt kombinálta a `p(w)` értékekkel.)

Összegzés

Az itt leírt módszereket olyan projektek használják a kéretlenlevél-szűrő feladatok teljesítményének jelentős növelésére, mint a Spambayes és a Bogofilter. A további fejlesztések, amelyek esetleg már a frázisalapú megközelítésre alkalmazzák a fenti módszereket, várhatóan még jobb teljesítményt mutatnak majd.

Linux Journal 2003. március, 107. szám

Gary Robinson
(grobinson@transpose.com)
A Transpose LLC cég vezérigazgatója. Személyes webnaplójában gyakorta tárgyal kéretlen levél témájú fejlesztéseket: radio.weblogs.com/0101454.

KAPCSOLÓDÓ CÍMEK

A cikkel kapcsolatos kérdések megvitatására, valamint az egyéb hírek és kéretlen levelekkel kapcsolatos ismeretek közreadásának külön Wikit szenteltünk ➲ http://www.wecanstopspam.org

➲ http://wecanstopspam.org/jsp/Wiki?GarySpamArticle

➲ http://www.paulgraham.com/spam.html

➲ http://www.bgl.nu/~glouis/bogofilter

➲ http://spambayes.sourceforge.net/index.html

➲ http://crm114.sourceforge.net

Az 1971-es Fisher hatékonysági tétel, ami a módszer leírásával egyetemben a következő munkában található: *R. Little* és *J. Folks* (1971) „Asymptotic Optimality of Fisher's Method of Combining Independent Tests”. Journal of the American Statistical Association, 66, pp. 802-806.

➲ http://research.microsoft.com/scripts/pubs/view.asp?TR_ID=MSR-TR-95-06

Bevezetés a FreeS/WAN használatába (2. rész)

Két saját hálózat biztonságos összekapcsolása már létező tűzfalrendszeren üzemelő FreeS/WAN-alagúttal.

Sorozatunk előző részében (Linuxvilág 2003. február, 37. oldal) a biztonságos VPN-ek – virtuális magánhálózatok – kialakításához szükséges IPSec alagútprotokoll-megvalósítást, a FreeS/WAN-t mutattam be. Mintaelrendezésként a távoli hozzáférésű (Remote Access – RA) VPN forgatókönyvét használtam. A rövidség kedvéért csak RA VPN-ként emlegetett hálózatban, emlékezzünk csak vissza, minden távoli felhasználótól elvárjuk, hogy saját kapcsolatot létesítsen a helyi hálózattal, ami így a „felhasználónként egy-egy alagút" elrendezést eredményezi.
De vajon mi történik, ha valamennyi felhasználónk ugyanahhoz a helyi hálózathoz (LAN) kapcsolódik? Az elmúlt hónapban már megemlítettem ezt a telephely–telephely közötti kapcsolattípust, de nem tértem ki felépítésének a részleteire. Éppen ezért kerül most a FreeS/WAN-nel kiépített telephely–telephely közötti kapcsolat figyelmünk középpontjába.

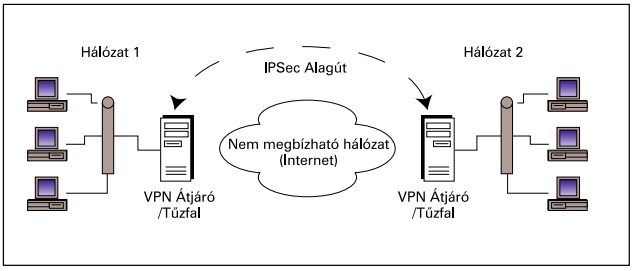
Kiépítés: telephely–telephely VPN-ek

Mielőtt azonban fejest ugranánk a FreeS/WAN beállítási lehetőségeibe, vessünk egy gyors pillantást a kiépítési megfontolásokra. Az *1. ábrán* egy jellegzetes telephely–telephely közötti VPN-elrendezést láthatunk.
Az 1. ábrán mindkét webhely tűzfala alagútvégpontként működik. Számos nyomós érv szól amellett, hogy a tűzfalat a VPN végpontjaként használjuk:

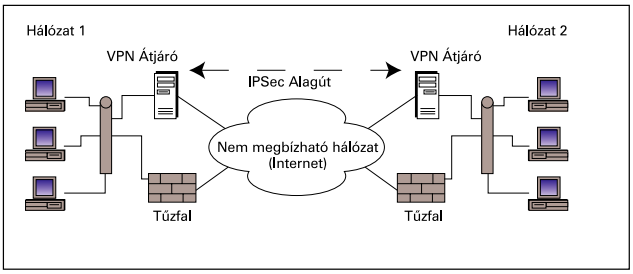
1. ***Kényelem***: a legtöbb tűzfalfelület támogatja az IPSec-et vagy valamilyen más VPN-protokollt, kiküszöbölve a különálló VPN-kiszolgáló gépek beállításához és felügyeletéhez szükséges időt és költségeket.
2. ***Biztonság***: a VPN-végpontként működő tűzfal nagyszerű szemcsézettséggel (granularity) és remek pontossággal képes szabályozni a forgalmat mind a VPN-alagútba belépő, mind az onnan kilépő irányba.
3. ***Egyszerűség***: ha a tűzfalalkalmazást és az IPSec-programot úgy tervezték, hogy ugyanazon a gépen fussanak, akkor könnyebben munkára lehet fogni az alagút-alkalmazásokat, ugyanakkor a hibakeresés is egyszerű, ha éppen nem hajlandók működni.

Vannak azonban olyan okok, amik miatt ez a megoldás nem vitelezhető ki jól, vagy csak egyszerűen nem kívánatos.

1. ***Együttműködés-képtelenség***: ha nem áll módodban az alagút mindkét végét ellenőrizni – mert például valamilyen gyártó- vagy partnerhálózathoz szeretnél kapcsolódni –, előfordulhat, hogy a távoli tűzfal VPN-megvalósítása nem képes a tűzfaladdal együttműködni.
2. ***Teljesítmény***: ha a tűzfalad a feladatok teljesítése közben már teljes mértékben vagy inkább túlságosan el van halmozva előfizetői kérésekkel (fully/over-subscribed), akkor lehetséges, hogy a többletterhet jelentő VPN-hitelesítést és -titkosítást már képtelen lesz támogatni.



1. ábra Egy egyszerű telephely–telephely közötti VPN-megoldás



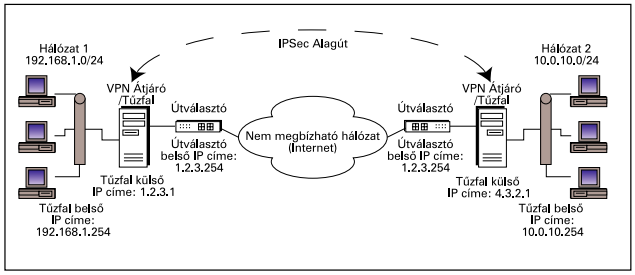
2. ábra Egy másik fajta telephely–telephely VPN-elrendezés

Ha éppen az iménti vagy más okok miatt a tűzfaladat nem lehet VPN-végpontként használni, akkor választhatod a *2. ábrán* bemutatott kiépítést.
A 2. ábrán bemutatott minden egyes számítógép kijelölt VPN-végpont (dedicated computer), és mindkét gép ilyen módon lett telepítve; a végpontok azonban eltérő felépítésűek is lehetnek, vagyis az egyik lehet tűzfal és VPN-végpont egyszerre, miközben a másiknál a feladatokat felosztják a gépek között. Gondatlanságnak tűnhet egy készüléket a tűzfallal párhuzamosan telepíteni. Vajon nem lehet-e egy ilyen módon beállított gépet hátsó ajtóként használni?
Dehogynem! Lehet, ha a VPN-kiszolgálógépet nem elég gondosan állították be, ezért az csak VPN-forgalmat hajlandó fogadni. A VPN program pedig olyan módon lett beállítva, hogy csakis ellenőrzött végpontokról, azaz erős hitelesítési módszerek használatával fogadjon kapcsolatokat. Akkor most vágjunk bele a FreeS/WAN-nal való munkába: lássuk, hogyan kell telephely–telephely VPN-t kiépíteni olyan végpontokkal, amik közvetlenül a tűzfalakra vagy önálló gépekre (standalone hosts) telepíthetők.

A példaforgatókönyv

A *3. ábrán* egy telephelyek közötti kapcsolatot magába foglaló VPN-kiépítés forgatókönyvét láthatjuk, ami működés szempontjából megfelel az 1. ábrának. Egyszóval ez azt jelenti, hogy mindkét telephelyen összetett tűzfal és FreeS/WAN-kiszolgálógép működik. De a *3. ábrát* böngészve ennél többet is megtudhatunk. Először is azt, hogy minden egyes hálózat helyi útválasztón keresztül kapcsolódik az Internethez. Másodszor az is

kiderül az ábrából, hogy az IP-címek az alagútmeghatározások elkészítéséhez szükségesek. Rövidesen kiderül, hogy melyik IP-címet hol kell felhasználni.
Ebben a forgatókönyvben a két telephely egy-egy tűzfalának külső csatolófelülete közötti VPN-alagutat kell kiépítenünk. Amikor az egyik telephely hálózatára bejelentkezett felhasználó érintkezésbe szeretne lépni a másik hálózatra csatlakozó valamelyik géppel, a tűzfal az alagúton keresztül elküldi neki a csomagokat. A válaszcsomagok ugyanúgy az alagúton ke-



3. kép A minta telephely–telephely kiépítésünk

resztül érkeznek vissza. Az alagúton keresztül mindkét oldalon levő gépek kezdeményezhetik kapcsolatok létrejöttét. Olyan több feladatot ellátó gépen, ami mind az IP Tables, mind a FreeS/WAN-kiszolgáló futtatására fel van készítve, ezek a tűzfalszabályok azonosak is lehetnek, mintha semmiféle alagutat nem használnánk, ráadásul ez a megoldás még NAT (Network Address Translation, azaz hálózati címváltás) használata esetén is megállja a helyét. De erre majd a cikk későbbi részében fogok részletesen kitérni.
Meg kell említeni néhány fontos előfeltételt erről a forgatókönyvről: elsőként azt, hogy mindkét tűzfal a 2.4.18-as Linux-rendszermagot használja. A tűzfalak rendszermagjának a foltozása az 1.97-es változatszámot viselő FreeS/WAN csomaggal történt, és a FreeS/WAN ugyanilyen változatszámot viselő, felhasználói terület kezelését végző eszközei ugyancsak telepítve lettek.
Harmadrészt a két hálózat képes tisztán, azaz IPSec nélkül elérni egymást. Természetesen nem kívánhatjuk, hogy ilyen módon folytassanak adatcserét, azonban tisztában kell lennünk a ténnyel, hogy vajon képesek lennének-e erre. Különben a VPN működését akadályozó tényezők felderítése sokkal nehezebb feladat lenne.

A gépkulcsok cseréje

E forgatókönyvben további fontos alapszabály az RSA-hitelesítés használata a „megosztott titok” helyett (shared secret). Miközben nem áll szándékomban, hogy az előző részbeli a gépi kulcsképzésről és karbantartásról szóló tudnivalókat újra elmondjam, fontos, hogy a lényegesebb részeket átismételjük. Remélhetőleg senki sem felejtette el, hogy minden egyes FreeS/WAN-t futtató gépnek egyedi kulccsal kell rendelkeznie. A FreeS/WAN bináris telepítő csomagjában található alapértelmezett kulcsot nem ajánlott használni. Ha azonban egy adott gépen létrehoztunk egy új kulcsot, akkor azt már annyi különböző alagútnál használhatjuk fel, ahányat az a bizonyos gép igényel. A gépre jellemző kulcs – ennek őrzése történik a */etc/ipsec.secrets* állományban – mindaddig hasznos, amíg rejtve marad, vagy a kódfejtő módszerek fejlődése miatt a kulcs túlságosan rövidnek bizonyul. Végeredményben ez az utóbbi lehetőség, mielőtt a FreeS/WAN maga is valamilyen ok miatt elavulttá válna, meglehetősen szerény.

A FreeS/WAN 1.92-es vagy ennél frissebb változatában a gépre jellemző egyedi kulcs hozható létre az alábbi parancs kiadásával:

```
# ipsec newhostkey --hostname my.host.fqdn
➥--output /etc/ipsec.secrets --bits 2192
```

A parancs 2192-bites RSA-kulcsot hoz létre, úgy, hogy mind a nyilvános, mind a saját alkotórészeket a */etc/ipsec.secrets* állományban helyezi el. Erre a múlt hónapban nem mutattam rá: mivel ezek a parancsok RSA-kulcsokkal dolgoznak, nagyobb kulcshosszok szükségesek, mint mondjuk az olyan tömbrejtjelezésnél, amilyen a 3DES.
Tehát még véletlenül se essünk kísértésbe, hogy 128, 196 vagy más háromjegyű számot próbálunk meg használni a `newhostkey --bits` parancsnál. Az olyan nyilvános kulcsképző módszerek, mint az RSA vagy a DSA, ezektől eltérően működnek, és azoknak nagyjából tízszer olyan hosszúaknak kell lenniük, mint a tömb- vagy folyamrejtjelező kulcsoknak (block/stream cipher keys). A legkisebb megfontolásra érdemes RSA-kulcsméret 1096 bit, de a 2192 bit sokkal biztonságosabb.
Az új nyilvános kulcs alagút-meghatározásba való másolásához, illetve beillesztéséhez használjuk a következő parancsot:

```
bash-# ipsec showhostkey --left
```

Értelemszerűen a `leftrsakey` utasítás helyett a `rightrsasigkey` létrehozásához a `--rightrsasigkey` utasítást használhatjuk. Ne felejtsd el, hogy ennek a parancsnak a kimenetét biztonságosan meg lehet osztani másokkal.
A gépre jellemző aláírás kulcsnak (host's signing key) csak a nyilvános részét tartalmazza. Akár titkosítás nélkül elküldhető levélben, elhelyezhető a weboldalon, vagy dalba foglalhatod a kedvenc kávézódban. Ez az, amiért az RSA-hitelesítés kényelmesebb, mint az osztott titok révén végzett hitelesítés, amiben biztonságosan és titkosan kell eljuttatni a hitelesítési bizonyítványt – osztott titkos jelsorozatot (shared secret string) – a másik helyre, amellyel IPSec alagutas kapcsolat kiépítését tervezed. Az RSA-hitelesítés még a „nemtörődömséget” is megengedi. Természetesen a */etc/ipsec.secrets* állomány kivétel ez alól, hiszen azt csak a rendszergazda számára szabad olvashatóvá tenni. Ezzel szemben a „osztott titok” által fémjelzett megvalósítás nem tűr semmiféle hanyagságot.

Az ipsec.conf telepítése

A FreeS/WAN beállítást végző legfontosabb állománya a */etc/ipsec.secrets* állományon kívül a */etc/ipsec.conf*. Az egyszerűség érdekében az alagút-meghatározások úgy lettek tervezve, hogy azok a FreeS/WAN-alagút mindkét végpontján ugyanolyan képet mutassanak. Ebből adódóan az itt következő példák programsorai a forgatókönyvünkben szereplő mindkét tűzfalon többnyire azonosak.
Az előző részben a figyelmemet főképpen az alagútmeghatározásokra irányítottam. Ezeket most is elő fogjuk venni, azonban először is mélyedjünk el a telepítési beállításban, illetve a `conn %default` szakaszokban. Az *1. lista* az egyik tűzfalunknak a telepítési beállításait mutatja – teljesen mindegy, melyiknek.
Az 1. listában szereplő első kapcsoló, az `interfaces` (csatolófelületek) fontos szerepet játszik: kijelöli a csatolókártyát, amellyel a gép a más IPSec-kiszolgálóktól érkező IPSec-kapcsolatokra figyel. Ez nem tévesztendő össze a csatolófelülettel, aminek révén a gép az alagúton keresztül küldött csomagokat figyeli. Ha az Internetre vagy más megbízhatatlan hálózatra

1. lista Alapbeállítás a /etc/ipsec.conf állományban

```
# Alap beÆll tÆs
config setup
        interfaces=%defaultroute
        klipsdebug=none
        plutodebug=all
        plutoload=%search
        plutostart=%search
```

2. lista Előre beállított értékek a /etc/ipsec.conf-ban

```
# AlapØrtelmezett ØrtØkek tovÆbbi kapcsolati
le rÆsokhoz
conn %default
        keyingtries=0
        disablearrivalcheck=no
        authby=rsasig
```

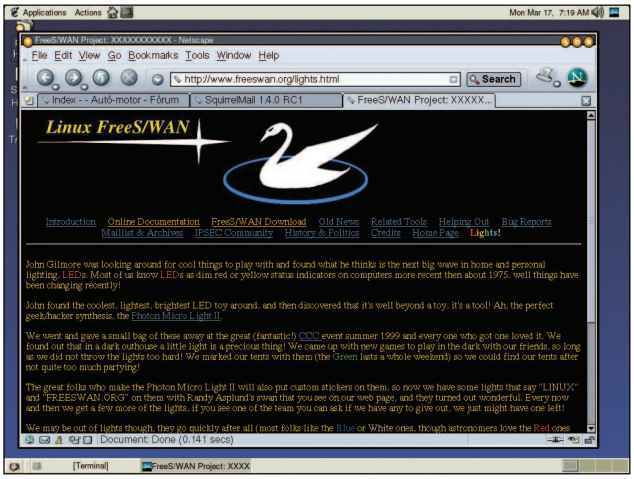
3. lista Alagút meghatározások a /etc/ipsec.conf-ban

```
conn brooks-reiner
    left=1.2.3.1
    leftsubnet=192.168.1.0/24
    leftnexthop=1.2.3.254
    # RSA 2192 bits
    # producers.brooks-sycophants.org
    # Mon Nov  4 06:38:32 2002
    leftrsasigkey=0obZZGi2UIghW... (
    right=4.3.2.1
    rightsubnet=10.0.10.0/24
    rightnexthop=4.3.2.254
    # RSA 2192 bits
    # plaidmen.reinerophiles.org
    # Sun Nov  3 23:01:39 2002
    rightrsasigkey=0sR4MLCTg5FJJ...
    auto=start
```

4. lista Az IPSec-et megengedő IP Tables-szabályok

```
# iptables -A INPUT -p udp -m udp --sport
500 --dport 500 -j ACCEPT
# iptables -A INPUT -p 50 -j ACCEPT
# iptables -A INPUT -p 51 -j ACCEPT
# iptables -A OUTPUT -p udp -m udp --sport
500 --dport 500 -j ACCEPT
# iptables -A OUTPUT -p 50 -j ACCEPT
# iptables -A OUTPUT -p 51 -j ACCEPT
```

külső felületként gondolunk, valamint a helyi hálózatra (LAN) a belső oldalként, akkor ennek megfelelően győződjünk meg róla, hogy az `interfaces` kapcsoló mindig a külső oldalra van irányítva.
A két hibaelhárítási lehetőség a `klipsdebug` és a `plutodebug` határozza meg a naplózás mennyiségét: a FreeS/WAN rendszermag csatolófelület démonja a `-KLIPS`, illetve az IKE-kulcsolás démonja a Pluto. Mindkét kapcsoló fogadja a mindent (`all`) vagy semmit (`none`) jelölő beszédes kulcsszavakat (magic words), valamint egy sor IPSec-tuljadonságot vagy eseményt, amik szintén naplóban rögzíthetők. A teljes felsorolás megtekintéséhez az ipsec_klipsdebug(8) és az ipsec_pluto(8) súgóoldalakat kell tanulmányozni.
A `plutoload` kapcsoló határozza meg, hogy mely alagútmeghatározásokat kell a FreeS/WAN indulásakor kezdőértékkel ellátni. A `%search` varázsszó (magic word) arra utasítja a Plutót, hogy minden további alagút-meghatározás `auto` kapcsolóját ennek megállapítására ellenőrizze, pontosabban minden alagutat, aminek a számára az `auto` lehetőség add-re van állítva. Hasonlóképpen a fentebb elmondottakhoz a `plutostart` mondja meg a Plutónak, hogy a FreeS/WAN indulásakor mely alagutakhoz kíséreljen meg önműködően kapcsolódni. Más szóval ez azt jelenti, hogy a `plutoload` arra utasítja a Plutót, hogy az alagutat a megadott gépek számára elérhetővé tegye; a `plutostart` viszont magának a Plutónak parancsolja meg, hogy „élesszen fel" bizonyos alagutakat anélkül, hogy a túloldali végpontokra várna. A `%search` érték megint csak megadható. Ebben az esetben ez meg fog egyezni azokkal az alagút-meghatározásokkal, amelyekben az `auto` kapcsoló `start`-ra lett beállítva.
A *2. listán* láthatjuk az ***ipsec.conf*** állomány `conn %default` szakaszát. A 2. listában az első kapcsoló, a `keyingtries` nullára van állítva, ami „korlátozás nélküli" jelentéssel bír. Ez azt jelenti, hogy amikor a Pluto megpróbál egy alagutat „felélesztení" vagy helyettesíteni, akkor annyiszor próbálja a kulcsot, ahányszor csak szükséges. Ez helyénvaló beállítás telephely–telephely típusú kapcsolat esetén, amelyben mindkét fél állandó kapcsolattal rendelkezik, de nem megfelelő távoli hozzáférésű VPN-ek esetén, amiben az ügyfelek csak szórványosan kapcsolódnak a hálózatra.
A `disablearrivalcheck` kapcsoló kikapcsolt állapotában arra kéri a KLIP-et, hogy az IPSec-alagútból a gépre lépő minden egyes csomagnál ellenőrizze, hogy a fejlécében rendelkezik-e hihető feladó és címzett IP-címmel. Az előre beállított érték „yes", tehát alapértelmezés szerint ezek az ellenőrzések nem történnek meg, éppen ezért kellene ezt „no"-ra állítani, hacsak nem pontosan tudod, mit teszel.
Végül az `authby` kapcsoló az, ami az alagutak számára az előre beállított hitelesítési módszert meghatározza, ez – mint azt már fentebb említettem – a mostani mintapéldánk számára az RSA (rsasig). Ezzel el is érkeztünk tényleges alagút-meghatározásunkhoz – mint azt a 3. ábrán láthatjuk.
Minthogy a mostani példánkban a telephely–telephely-forgatókönyv megvalósítást tárgyaljuk, a FreeS/WAN szokásos jelölésrendszere – kiszolgáló = left (bal), távoli hozzáférésű ügyfelek, RA clients = right (jobb) – most nem bír jelentéssel. Így ebben az esetben teljesen önkényes, hogy melyik oldalt tekintjük a jobb oldalnak, s melyiket a balnak. Az egyetlen fontos teendő, hogy az alagútmeghatározások kialakítását mindkét gépen következetesen és összhangban kell végrehajtani. A példában – 3. ábra – az Internettől balra eső webhelyet a bal oldalnak (left) tekintjük, miközben az Internettől jobbra eső telephelyet a jobb oldalnak (right). Ez teljesen magától értetődőnek hangzik, de ha az oldalakat fordítva jelölném ki, az alagút akkor is a fentebbi módszerrel megegyezően működne, feltéve, hogy mindkét oldalon ugyanazokat a beállításokat használom.
Mint az a 3. ábrából kitűnik, a bal oldal (left) a külső csatolófelü-



let IP-címére van beállítva, vagyis az Internet számára elérhető, az alagút pedig várakozó állapotban van. A `leftsubnet` azonban olyan hálózati címre lett beállítva, ami fogadja a bejövő csomagokat, mármint az alagutat elhagyó csomagokat.
A `leftnexthop` az Internet és a tűzfal/IPSec-gép közötti következő ugrás (next hop) IP-címe; és természetesen a `leftrsasigkey` a bal oldali gép kulcsa. Következtetési képességedre bízom, hogy kitaláld, a 3. ábrán mely gépekre illenek leginkább ezek a jellemzők.
Aztán itt van még az alagút `auto` kapcsolója, ami `start`-ra van beállítva. Amikor a Pluto indulásakor az alagút-meghatározásokkal kapcsolatos utasítások megkeresését végzi, ez a beállítás írja elő, hogy az alagút kiépítése a fentebbi meghatározás alapján kezdődjön el.
Ebben a forgatókönyvben korábban már felhívtam a figyelmet rá, hogy a */etc/ipsec.conf* állományok a tűzfalak és átjárók esetén azonosak. Ha már egyszer telepítve vannak, akkor minden gépen elindíthatjuk az IPSec-et, és megkezdhetjük az alagutazást. A legtöbb Linux-változatban ezt az alábbi parancs kiadásával tehetjük meg:

```
bash-# /etc/init.d/ipsec start
```

Amennyiben az IPSec már fut, ezt a parancsot használd:

```
bash-# /etc/init.d/ipsec restart
```

Ha az IPSec már el vagy újra lett indítva, az alagút működésbe lép, és útválasztás révén a másik hálózathoz irányítja a forgalmat az alagúton keresztül. Ez az útválasztás önműködően történik, vagyis a `leftsubnet` és a `rightsubnet` kapcsolók alapján, a */etc/ipsec.conf* alagút-meghatározás állományban leírtak szerint.

A tűzfalak és a NAT

Nyilvánvaló, hogy korlátozni szeretnéd azokat a tevékenységeket, amiket a másik hálózat gépei tehetnek a te hálózatodon, és fordítva. Már korábban említettem, a FreeS/WAN-t futtató linuxos gépen a tűzfalszabályok nem feltétlenül térnek el az alagút nélküli szabályoktól. Ez még a NAT használata mellett is igaz. Minden egyes átjáró számára elkészítjük a tűzfalszabályokat, beállítjuk a `FORWARD`, `PREROUTING` és `POSTROUTING` szabályokat pontosan ugyanúgy, mintha nem is használnánk IPSec-et – épp csak a csatolófelületekkel kell nagyon óvatosan bánnunk. Ha a `-i` és `-o` kapcsolókat használod, még véletlenül se használd az `eth0` jelölést, ha közben az `ipsec0`-ra gondolsz, vagy az `ipsec+`, ha pedig az „összes alagút csatolófelületre”! Ha kétségek merülnének fel, a tűzfalszabályokban inkább ragaszkodj az IP-címekhez, mint a csatolófelületek elnevezéséhez.
Ezenkívül győződj meg róla, hogy NAT-műveletek nem hajtódnak végre az alagúton átküldött csomagokon. Az IPSec-csomagok fejlécének ellenőrző összege (checksum) megtalálható a csomag adatmezőjének testében. Az IP-fejléc újraírása – például a forrás vagy cél IP fordítása révén – megsérti ezt az üzenetkivonatolást (message digest), és ebből furcsaságok adódhatnak. Hálózati címváltás (NAT) végezhető olyan csomagokon, amik elhagyják az alagutat vagy éppen belépnek oda, de semmiképpen sem az idő alatt, amíg benn vannak a folyamatban.
Bármi mást teszel, mindkét átjárón legalább három újabb szabályra lesz szükséged, hogy megengedd az IPSec-kulcscsegyeztetést és alagutazást. A bemenet- és kimenetláncokon (Input/Output chains) engedélyezni kell az UDP 500-as kapu számára küldött IP-protokollú 50-es és 51-es csomagokat.
A mindkét átjáróra vonatkozó szabályokat tükrözik a *4. listán* láthatók.

Összegzés

A fentebb elmondottak segítségével hálózatodat biztonságosan és olcsón összekötheted gyártóid, üzleti partnereid, valamint ismerőseid hálózatával.
Sok sikert!

Linux Journal 2003. február, 106. 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 With Linux”.

KAPCSOLÓDÓ CÍMEK

A FreeS/WAN hivatalos honlapja
➲ http.//www.freeswan.org. Forráskódok, hírek és rengeteg hálózatról elérhető leírás lelőhelye.
A /etc/ipsec.conf súgóoldala: ipsec.conf(5).

Coyote Linux: egylemezes útválasztó otthonra

Közismert, hogy mind a linuxos, mind az újabb windowsos gépek segítségével internetkapcsolatunkat megoszthatjuk. De kissé túlzás lenne erre a feladatra egy teljes munkaállomást használni.

A feladat megoldható egy kisebb teljesítményű, „maradék" alkatrészekből összeállított géppel és valamely kisméretű Linux-terjesztés segítségével létrehozott útválasztóval. Manapság alapvető igény, hogy a számítógépünkkel csatlakozni tudjunk az Internetre. A Linux, mivel az egyéb Unix-változatokhoz hasonlóan alapvetően hálózati rendszer, természetesen képes erre. Ennek ellenére a szolgáltatók ezt nem mindig veszik figyelembe, és amikor lehetővé vált, hogy nálunk is bekössék az ADSL-csatlakozást, kissé aggódtam is, hogy mennyi gondom lesz majd vele, mire beüzemelem Linux alatt. Elhatároztam, hogy leírom a folyamatot, hogy másokkal is megosszam a tapasztalataimat.

Az ADSL-történet

1.

Nos, ebből nem lett semmi. Amikor a szerelők megjelentek, még igazolódni látszott az elképzelésem. Kissé megrettentek ugyanis, amint megemlítettem, hogy jó lenne linuxos gépre beállítani az elérést. Már látták magukat, amint órákon át küzdenek hiába, ahelyett, hogy a CD-n mellékelt meghajtót Windowsra feltelepítve a szokásos módon járnának el.
Némileg bonyolította a helyzetet, hogy nem is tudtam előre, milyen módon fog a gép a szolgáltató gépéhez csatlakozni. Erre a feladatra két elterjedt változat is ismeretes: a DHCP-híd, valamint az úgynevezett pppoE, azaz ppp ethernet (ppp over ethernet). Az utóbbi a gyakoribb, de a szerződésben, illetve az ahhoz mellékelt adatlapban olyasmik szerepeltek, amikből arra lehetett következtetni, hogy DHCP-ről lesz szó…
Az ADSL-eléréshez két eszközt szereltek be. Az egyik ahhoz szükséges, hogy egy időben lehessen telefonálni és az ADSL-es internetelérést használni. Ez egy úgynevezett Splitter (*1. kép*), lényegében egy szűrő, ami szétválasztja a nagyfrekvenciás (ADSL Internet) és a kisfrekvenciás jeleket, amelyek ugyanazon az érpáron át érkeznek (telefonvonal).
A másik maga az ADSL-modem, amelynek több fajtája is létezik. Ha Linuxon is szeretnénk használni, célszerű azt kérni, amelyik az ADSL-kimenet mellett egy ethernetcsatlakozóval bír (UTP). Emellett lehetséges az USB-változat is, de ezt kerülném, mert a Linux meghajtói nem annyira megbízhatóak, valamint a kisméretű Linux-terjesztések esetében fordítási gondjaink adódhatnak (vagy legalábbis fel kell telepítenünk egy olyan változatú rendszermagot, mint a célrendszeren futó). Az előzetes leírással ellentétben nem Alcatel, hanem Cisco modemet hoztak (*2. kép*).

2.

Így felkészülésem szinte hiábavaló volt, ugyanis az Alcatel készülék adatlapját töltöttem le.
Miután a szerelők a magukkal hozott (windowsos) hordozható géppel és általános belépési kódjukkal kipróbálták a kapcsolatot, az ethernetkábel végét átdugtuk a linuxos gépbe. Ezen SuSE 8.1 fut. És jött a meglepetés: a felhasználónév és jelszó beírása, valamint a névkiszolgáló beállítása, és a pppoE-mód kiválasztása után a rendszer azonnal működött. Semmit nem kellett még telepíteni sem, nemhogy újraindítani... (Ezek a beállítások a Yast program ***Hálózatalap/ADSL beállítása*** lapján találhatók).
Így ez a téma aligha lenne elegendő egy cikkre, viszont még aznap beállítottam egy útválasztót, hogy a most már rendelkezésre álló gyors kapcsolatot ki tudjuk használni. A mi családunkban ugyanis az egy főre jutó számítógépek száma messze meghaladja a magyar, de talán még az amerikai átlagot is, még akkor is, ha a kutyát is beleszámolom. Felmerül a kérdés, hogy miért kell külön útválasztó, amikor a fent említett és más Linuxok, de az újabb Windowsok is biztosítják a kapcsolatmegosztást. Tehát az adott gépnek mindenképpen működnie kell, ha valaki Internetet akar használni, ezenkívül a munkaállomásokban általában nincs túl sok szabad bővítősín a további hálózati kártyáknak, azonkívül a belső hálózatot valamilyen mértékig célszerű az Internetről leválasztani.

Útválasztó

A továbbiakban a linuxos útválasztó kialakításához a sok lehetőség közül az egyik megoldást ismertetem, ami a Coyote Linux-változaton (➲ http://www.coyotelinux.com) alapul. Esetében egy a Linux Router Project alappilléreire épülő, rendkívül kis helyigényű Linuxról van szó. A Vortech cég dolgozik rajta üzleti célú fejlesztéseivel párhuzamosan,

az alapváltozata ingyenes. A cég teljes útválasztót (eszközzel együtt), valamint tűzfalat is készít.
Mik azok a tulajdonságok, amiket egy útválasztótól elvárnánk? Miután ezt tisztáztuk, meg tudjuk állapítani, hogy a választott terjesztés teljesíteni tudja-e az elvárásainkat. Lássuk őket!

- Kis gépigény: ilyen módon régi, „levetett” gépünket újra munkára bírhatjuk, és az egész alig kerül valamibe.
- Megbízhatóság: nem szerencsés, ha sokszor kell vele foglalkoznunk – a ki- és bekapcsoláson kívül.
- Egyszerű használat: működtetése ne igényeljen külön programokat a munkaállomásokon sem.
- Elterjedt (olcsó) alkatrészek támogatása.
- Különböző internetelérési módok, valamint a belső hálózat támogatása.

Mint várható volt, ezeket a feltételeket a Coyote teljesíti.
A következőkben megnézzük, hogyan tudjuk felhasználni céljaink eléréséhez. Előbb azonban soroljuk fel, mire is lesz szükségünk:

1. Az internetelérés valamilyen formájára: modemes behívó kapcsolat, a ISDN-es behívásra és ADSL-re. A rádiós kapcsolat esetén nem biztos, hogy a Coyote alapterjesztés tartalmazni fogja a rádiós hálózati eszköz meghajtómodulját. ISDN esetén, csakúgy mint a modemnél, ha tehetjük, külső eszközt válasszunk. Cégemnél ez sikerült is, a szolgáltató egy Multi-Terminal nevű eszközt bocsátott rendelkezésünkre (*3. kép*).

3.

Ebben beépített ISDN-telefonkészülék-átalakító, valamint USB-s és soros csatlakozás modem is van. A soros modem Linuxból is remekül működik.

2. Egy megbízhatóan működő régi számítógépre. Elvileg elég lehet egy 386 DX is, de ez csak igazán lassú kapcsolatnál nem jelent szűk keresztmetszetet, egyébként egy 486 DX alaplap teljesen elegendő.
3. Legalább 8 MB memória szükséges, de 12 MB az ajánlott. A modemes kapcsolat esetén egy 486 DX2-50-es alaplapot tudtam használni 8 MB memóriával, igaz, így csak a szükséges alapösszetevőket lehetett feltenni az útválasztóra.
4. A következő kapcsolatnak megfelelő eszközök szükségesek hozzá: modem és soros ISDN esetén soros kapu, ADSL és kábel esetén pedig ethernetkártya.
5. A belső hálózatnak megfelelő eszközre (célszerűen és többnyire): az ethernetcsatolóra.
6. Egy megbízható 3 1/2"-os hajlékonylemez-meghajtóra. Sem merevlemez, sem CD nem szükséges.
7. A belső hálózat kiépítése a következőképpen néz ki: hálózati kártya a munkaállomásokba, a szükséges kábelek, elosztó (hub) vagy kapcsoló (switch). Koaxiális kábel esetén az utóbbi kettő nem feltétlenül kell.
8. A telepítéshez egy videokártya is ajánlatos. Bármilyen megteszi, amihez akad monitorunk. Telepítés és kipróbálás után csak akkor lesz rá szükségünk, ha a BIOS nem hajlandó nélküle elindítani a gépet, mivel a működő útválasztóra távolról is be tudunk majd jelentkezni. Szintén csak a telepítés ideje alatt szükséges egy billentyűzet. (Addig akár az egyik munkaállomásról is levehetjük.)
9. Szükség van még a Coyote Linux telepítőkészletére.
10. Nem utolsó sorban egy működő Linux-rendszerrel is rendelkeznünk kell, amelyen az útválasztó hajlékonylemezét elkészítjük.

A Coyote Linux

A megadott honlapról töltsük le a Linux alapú, hajlékonylemezes változatot: ***Coyote Linux/Downloads/Floppy release/Linux floppy creator scripts v 1.32***, vagy ha van, akkor az újabbat. (Windowsos beállítóprogram is létezik, de a Linux alattival több szolgáltatást érhetünk el, és mint régi linuxosoknak, amúgy sincs Windowsunk).
Ezekből a parancsfájlokból készítettem egy nem hivatalos magyar változatot, amit úgy használhatunk, hogy a letöltött `abdesign.tar.gz` állományt kibontjuk (ez egy ***coyote*** alkönyvtárat hoz létre), majd az általam mellékelt `coyote.hu.tgz` állományt kicsomagoljuk a frissen létrejött ***coyote*** könyvtárban. Ekkor az felülírja az eredeti angol parancsállományokat.
Ezzel az előkészületeket gyakorlatilag be is fejeztük. Még elő kell vennünk egy jó minőségű hajlékonylemezt (ma már ez a feladat egyik legnehezebb része), és neki is kezdhetünk az útválasztó készítésének.

Az útválasztó kiépítése

Logikusnak tűnne előbb az útválasztó fizikai kiépítése. Ezt is megtehetjük, hogy megnyugtassuk magunkat, valóban működik. Ha a BIOS képernyő megjelenik, az már biztató jel Esetleg egy régi DOS-os hajlékonylemezzel kipróbálhatjuk a rendszerindítást is. (Akit a Linux mellett az ingyenes, vagyis GPL DOS is érdekel, az nézze meg a ➲ http://www.freedos.org címen. Próbacélokra kitűnő és felidézi a „régi szép” DOS-os időket. Emellett kiváló fejlesztőkörnyezet is létezik hozzá: egy védett módú futtatórendszerrel ellátott gcc-változat, djgpp, valamint egy fejlesztői felület, az rhide.) Ha mind a belső hálózatunk, mind az internetelérésünk ethernet, illetve több belső hálózattal rendelkezünk, mind(egyik) ethernetillesztéssel, akkor a hálózati kártyák beépítésével még várjunk. Az eredeti Coyote-leírás azt ajánlja, hogy több ethernetcsatolóhoz ne használjunk azonos típust. Valójában ezzel nincs semmi baj, az alábbiakban le fogom írni, miként járjunk el, hogy ne legyen gond és ne keverjük össze a kártyákat (az én útválasztómban három azonos típusú kártya van).
Ezután kezdjük el a rendszer programrészének elkészítését: ha megvan a megbízható hajlékonylemez, indíthatjuk is. (Aki kissé bizalmatlanabb típus – hajlékonylemezek esetén az ember nem lehet elég bizalmatlan, csinálhat egy próbaformázást.) A Coyote egy kissé túlformázott lemezt ajánl, ezt így próbálhatjuk:

```
fdformat /dev/fd0u1680
```

Ha az ellenőrzés is hibátlanul lezajlik, akkor jó esélyünk van arra, hogy a Coyote rendszer is használni tudja majd. Lépjünk be a Coyote könyvtárába. Indítsuk el a `makefloppy.sh` parancsfájlt. (Ha a fenti formázás felhasználóként működött, akkor a folyamat további része is menni fog. Ha nem, akkor ezt is rendszergazdaként kell tennünk.) Ha mind az eredeti Coyote-archívumot, mind az én módosító állományomat jól csomagoltuk ki, akkor a következő lépéseken kell végighaladnunk.

Ali baba és a negyven lépés

1. A készítendő hajlékonylemez méretének megválasztása. Az előbbi próbának megfelelően célszerű az 1,68 MB-os változatot választani. Ez gyakorlatilag minden lemezmeghajtóval működik, és ráférnek a távoli karbantartást biztosító csomagok is. Írjuk be: 2, majd nyomjunk ENTER-t.
2. A célrendszer processzortípusának a kiválasztása. A kérdés gyakorlatilag a matematikai processzor meglétét vizsgálja. Ha sikerült egy 486 DX alaplapot szereznünk, akkor (2).
3. Az internetkapcsolat típusának a beállítása. Itt a kapcsolat típusától függően a további kérdések különböznek egymástól. Előbb megnézzük a modemes (és külső ISDN-es, mert az ugyanúgy működik) kapcsolatot, utána az ADSL/kábelmodemes (pppoE) változatot. A behívásos kapcsolathoz válasszuk a (3) pontot.
4. A rendszer beállítása ppp behíváshoz. Először a helyi hálózat alapértelmezett beállításait láthatjuk. Ha nincs külön „kiforrott" tervünk a belső hálózat IP-címeit illetően, akkor célszerű ezt elfogadni. (Ez egy C osztályú, úgynevezett nem nyilvános IP-címtartomány). Válasszuk az N-t az elfogadáshoz!
5. Ezután az igény szerinti tárcsázás lehetőségét kell meghatároznunk. Ha nem engedélyezzük, akkor már a bekapcsolás után megpróbál csatlakozni. Az engedélyezéshez nyomjunk I-t.
6. Tétlenségi kapcsolatbontás. Ha egy adott ideig nincs adatforgalom, bontsa-e a kapcsolatot (mondjuk ottfelejtjük a gépet, és ketyeg a telefonszámla…). Ismét nyomjunk I-t az engedélyezéshez.
7. A tétlenségi idő – ami után bont – hossza másodpercben. Az alapértelmezett idő 180 másodperc , nyomjuk le az ENTER billentyűt, ha elfogadjuk – vagy írjunk be más értéket.
8. Állandó IP-címünk van? Modemes kapcsolatnál nem valószínű, ezért N.
9. Meg kell adni egy IP-címet a ppp-csatolónak. Ezt valójában nem fogja használni, csak azért van rá szüksége, mert indításkor is IP-címmel kell bírnia. A kapcsolat létrejötte után a ppp démon ezt a valós dinamikus címre módosítja. Fogadjuk el a felkínált értéket.
10. Melyik soros kapuhoz kapcsolódik a modem? Az alapértelmezett a `ttyS0` (COM1). Ha másikhoz kapcsolódik, akkor írjuk át (pl.: `ttyS1`, azaz COM2).
12. A soros kapu sebessége. Általában jó a megadott 11 5200 Baud. Ha régi illesztőt használunk, esetleg lejjebb kell vennünk. Tapasztalatom szerint ezt minden AT soros illesztő bírja, de lehetnek kivételek. A 486-os alaplapokra építettek biztosan bírják ENTER.
13. Modemindító (alaphelyzetbe állító) karaktersor. Ha nincs róla tudomásunk, hogy modemünk igényel valamit, akkor fogadjuk el. Modemszakértők esetleg némító parancsot írhatnak ide ENTER.
14. A szolgáltató nevének megadása. Csupán a későbbi azonosításra szolgál – írjunk be egy rövid, de sokatmondó nevet.
15. A hívandó telefonszám megadása, például 06 51 690690. Ez a szolgáltatótól függ, és szerződéskötéskor kapjuk meg.
16. Felhasználónév megadása. Ezt ugyancsak a szolgáltatótól kapjuk vagy vele kell egyeztetnünk. Írjuk be, ügyelve a kis- és nagybetűkre.
17. A jelszó megadása az előbbiekhez hasonlóan. Ha a jelszó beírásakor úgy tűnik nekünk, hogy megbízhatatlan személyek is lézengenek körülöttünk, a begépelést halasszuk későbbre!
18. Kell-e azonosítani magunkat a ppp futtatása előtt? Nálunk kellett (I) az, de még mielőtt rákérdeznének, jobb nélküle kipróbálni.
19. Döntenünk kell, hogy meg kell-e adnunk a Coyote DHCP-kiszolgálót; továbbá hogy a belső hálózaton állandó vagy dinamikus IP-címeket szeretnénk-e használni. Ha kevés gépünk van, célszerűbbnek tűnik az állandó IP-címek választása (N).
20. Tartománynév választása. A mi esetünkben (álcázott belső IP-címek) bármi lehet, például ***otthon***.
21. Az első névkiszolgáló IP-címének megadása. Ezt is a szolgáltatónktól kapjuk. Ha mégsem, esetleg próbáljunk meg egy teljes Linuxból, a tárcsázóprogrammal kapcsolódni (SuSE alatt `wvdial`), ez ugyanis kiírja a kapott névkiszolgáló-címet – használjuk fel azt.
22. A második névkiszolgáló begépelése. Ha tudunk, még egyet megadhatunk…
23. El kell döntenünk, hogy telepíteni akarjuk-e az `sshd` kiszolgálót a biztonságos távoli eléréshez (I). Ha igen, elkészíti a kiszolgálóoldali `ssh`-kulcsokat és a gépazonosítót.
24. Ki kell választanunk, hogy telepíteni akarjuk-e a `webctl` webes felületű karbantartó csomagot. (Nem működik egészen jól, de adatforrásként jó, például a pillanatnyi IP-cím lekérdezéséhez a munkaállomásokról.) Nyomjunk ismét I-t.
25. A hálózati kártya modulnevének a beírása. Nem a gyártó az érdekes, hanem a rajta lévő lapkák. A legtöbb olcsó 10 Mbit-es kártya az `ne2k-pci` modullal működik, a 100 Mbit-esek az `rtl8139`-cel. (Például a Realtek-lapkás kártyák). A régebbi ethernetkártyák jelentős része a `ne` meghajtóval működik. Gyakori még az SMC Ultra kártya, mely az `smc-ultra` vagy `smc-ultra32` meghajtóval megy. Ha PCI-sínes kártyáról van szó, a kapcsoló szükségtelen, egy ISA-buszosnál azonban nélkülözhetetlen (a minta szerint például: `ne io=0x300 irq=10`). A megfelelő kapcsolót próbáljuk meg a kártya beállítóiról leolvasni, vagy ha működött valamelyik rendszeren, akkor onnan.
26. Ezután tegyük be a hajlékonylemezt, és már el is készülhet.

Nézzük az ADSL változatot!

Az ADSL változat a 3. ponttól kezdődően tér el a fentiektől.

3. A `pppoE` módhoz válasszuk a (2.) pontot.
4. A fentihez hasonló a helyi hálózat beállítása is. Hagyjuk meg a felajánlott értékeket (N).
5. A felhasználónév beírása: magától értetődő.
6. A jelszó begépelése: szintén.
7. A tartománynév megadása: mint fent, bármi lehet.
8. Az első névkiszolgáló IP-címének a megadása – mint fent.
9. A második névkiszolgáló megadása ugyanúgy történik.
10. Választanunk kell: állandó vagy szükség esetén felépített és bontott kapcsolatot szeretnénk-e. Ha ADSL-ünk átalánydíjas, akkor választhatjuk az állandót, ekkor csak az útválasztó vagy a modem kikapcsolásával, illetve a szolgáltató bonthatja a kapcsolatot (technikai okokból – ilyen esetben az útválasztó újra megpróbálja létrehozni

Egy egyszerű példa a MAC-listára

```
ne2k-pci.c: v1.02 for Linux 2.2, 10/19/2000, D. Becker/P. Gortmaker,
http://www.scyld.com/network/ne2k-pci.html
ne2k-pci.c: PCI NE2000 clone 'RealTek RTL-8029' at I/O 0x6000, IRQ 9.
eth0: RealTek RTL-8029 found at 0x6000, IRQ 9, 00:50:BF:6E:9C:9A.
ne2k-pci.c: PCI NE2000 clone 'RealTek RTL-8029' at I/O 0x6100, IRQ 10.
eth1: RealTek RTL-8029 found at 0x6100, IRQ 10, 00:20:18:2A:7B:D7.
ne2k-pci.c: PCI NE2000 clone 'RealTek RTL-8029' at I/O 0x6200, IRQ 11.
eth2: RealTek RTL-8029 found at 0x6200, IRQ 11, 00:20:18:2A:61:58.
```

a kapcsolatot). Ha idő- vagy forgalmi díjunk van, akkor választhatjuk a modemes kapcsolathoz hasonló önműködő csatlakozást és bontást (állandó: 1).

11. Szükséges-e az `sshd` kiszolgáló a távoli eléréshez? Igen, azaz nyomjunk I-t.
12. Működjön-e az útválasztó DHCP-kiszolgálóként? (Állandó belső IP-címek esetén: N.)
13. Az internetes és helyi hálózati kártya modulneve és esetleges kapcsolóinak a megadása. Ha különböző típusú kártyáink vannak, akkor már itt eldől, hogy melyik melyik lesz. Ha egyformák, akkor ezt a későbbiekben határozzuk meg.
14. Döntenünk kell, hogy szükségünk van-e az `Oidentd` csomagra. Hagyományos internetezéshez még nem volt rá szükségem… (N).
15. Választanunk kell, hogy szükségünk van-e a `webctl` webes felületű karbantartócsomagra? (I)

Az eredményt máris kiírhatjuk Coyote-rendszerlemezünkre.

Az első próbák az útválasztóval

Az első próba alkalmával mindjárt meg is állapíthatjuk, hogy melyik hálózati kártyánk melyikként fog működni. Helyezzünk a gépbe egy kártyát, valamint hajlékonylemezt, majd indítsuk el a gépet. Türelem, a Coyote a hajlékonylemezes egység lassúságának megfelelő tempóban indul el. A rendszerüzenetek között figyeljük meg a hálózati kártyáról szólót (ha esetleg nem látszana, a bejelentkezés után pedig elindult az útválasztó beállítóparancsfájlja, akkor a ***q*** pontot választva lépjünk ki, és a `dmesg` paranccsal kérjük le újra a rendszerüzeneteket).
A kártya csatlakozója mellé ragasszunk címkét, de filctollal akár közvetlenül a fémre is írhatunk.
Egy papírra jegyezzük fel az első kártya ethernetcímét (MAC) a listából. Ha ráfér, mindjárt a kártyán lévő címkére is írhatjuk (pl.: 00:50:BF:6E:9C:9A). Ezután ugyanígy járjunk el a második, illetve ha több belső hálózatot tervezünk, akkor a harmadik kártyával is (az előző kártyát nem kell kivennünk, egyszerűen figyeljük, hogy melyik az új cím). Végül a *listánkon* láthatóhoz hasonlót kapunk az újraindítás után.
Ezután a MAC-címek alapján, a Coyote `eth0`, `eth1` stb. számozással már össze tudjuk rendezni az általunk megszámozott kártyákat. A kártyára a csatlakozó mellé rá is írhatjuk, hogy melyik csatolóként működik. (Modemes kapcsolat és egy belső hálózat esetén ez a rész kimarad, mivel ott egyértelmű a dolog).
Csatlakoztassuk a hálózatokat! Az alapértelmezett belső hálózatot az `eth0`-csatolóhoz, az Internetet a megfelelő soros kapun vagy az `eth1`-csatolón át a megfelelő modemhez kapcsoljuk. Indítsuk újra az útválasztót. A helyi hálózat egyik gépéről próbáljuk meg `ping` segítségével elérni.

```
ping 192.168.0.1
```

Ha választ kapunk, akkor a belső hálózat működik. (Feltételeztem, hogy a belső hálózat gépeit beállítottuk a megfelelő hálózaton való működéshez, valamint hogy az alapértelmezett IP-címeket használjuk). Ha mégsem, akkor a következő beállításokat használjuk:

- csatlakozás az Internethez helyi hálózaton keresztül,
- IP-cím 192.168.0.xxx (az xxx gépenként más: nem lehet 0, mert az maga a hálózat, 255 sem, mert az a csoportos adatküldés (broadcast) címe, valamint 1 sem adható meg, mert az maga az útválasztó),
- hálózati maszk: 255.255.255.0; és az átjáró (gateway): 192.168.0.1.

Mind a linuxos, mind a windowsos gépeket ugyanilyen módszerrel kell beállítani. A fent megadott névkiszolgálót célszerű ugyancsak azon nyomban beírni.
Ezután csatlakoztassuk a modemet (ADSL vagy analóg) a telefonvonalhoz, és kapcsoljuk be. Esetleg újraindíthatjuk az útválasztót (elvileg nem szükséges, de erről a módszertől már világhírűvé vált egy-két operációs rendszer…). A belső hálózat valamely gépéről próbáljunk meg IP-cím szerint elérni egy külső gépet. (Ha más nem jut eszünkbe, a névkiszolgálót is próbálhatjuk. Ez nem biztos, hogy fog válaszolni rá, mivel sok esetben le van tiltva, de az útválasztó ezt előre nem tudhatja, és felépíti a kapcsolatot).
Közben jelentkezzünk be a coyote-os gépen rendszergazdaként. (Először nincs jelszó, amit később be kell állíthatunk.) Önműködően elindul az `lrcfg` nevű, karbantartó parancsfájl. Ennek menüjéből (angol) válasszuk a *c* pontot, ekkor kijelzi az időszerű állapotot. Ha az „internet configuration" szöveg alatt valós IP-címet látunk, akkor a kapcsolat működik (hagyományos modemnél ezt a szokásos „összefütyülés" előzi meg; ADSL és ISDN esetén pedig szinte azonnal létrejön a kapcsolat).
Ha valós IP-címet látunk, egy belső gépről próbáljuk meg név szerint elérni kedvenc weboldalunkat. Ha sikerül, akkor van egy működő útválasztónk!
Az esetleges hibák javításáról és a másik helyi hálózat beállításáról, valamint az útválasztó finomhangolásáról, továbbá a távoli karbantartásról a következő részben lesz szó.

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

Két munkaállomás egy gépből

Mindenki számára nyilvánvaló tény, hogy a GNU/Linux az egyik legnyitottabb és legjobban testre szabható operációs rendszer, ezért is kedvelik annyira a kiszolgálóknál, a beágyazott rendszereknél és még ezer helyen.

Hosszú ideje foglalkoztat az a kérdés, hogy meg tudjuk-e oldani Linuxszal azt a látszólag egyszerű feladatot, hogy egyetlen gépre több monitort, billentyűt és egeret kötve egymástól független munkaállomásokat alakítsunk ki. Mert ugyebár az eszközöket csatlakoztatni tudjuk, főleg mióta olyan szabványok terjedtek el, mint például az USB. Mielőtt belekezdenénk, érdemes beleolvasni a A többképernyős Linux-rendszerek című cikkbe (Linuxvilág 2002. augusztusi szám, 41. oldal).

Miért jó ez?

Mi hasznunk származik abból, ha egy asztali gépből többmunkaállomásos kiszolgálót fabrikálunk? Az irodai hálózatok legnagyobb baja, hogy külön kell őket kezelni, minden egyes gépen külön kell frissíteni a csomagokat, beállítani a programokat és így tovább. Bár használhatunk egy központi felügyeleti rendszert, ez gyakran az „ágyúval verébre lövés” esete. Tegyük fel, hogy egy helyen hat munkaállomást szükséges létrehoznunk, a költségek alacsonyan tartásával, például két szobában, ahol a gépek ráadásul egymás mellé kerülnének. Erre lehet megoldás egy kiszolgáló és hozzá kapcsolódó tetszőleges számú, csupán grafikus felületet biztosító gyengébb gép hálózata is. Hiszen mennyibe kerül manapság egy buta, öreg Pentium I-es, amire felhúzunk egy alapkiépítésű XFree86 rendszert, majd a megfelelően beállított kiszolgálóról egy egyszerű `X -query kiszolgÆl` paranccsal már meg is oldottuk a dolgot. Igen, a módszer jól működik, ha van megfelelő helyünk a kis gépnek, ha nem gond a gép karbantartása, és nem utolsó sorban nem zavar a plusz egy gép zörgő-csörgő zaja. Bár egy ilyen kis gép kialakítható valamilyen profibb kivitelű, például mini ATX-es gépből, ezeknél már komoly kérdés az ár.
Tehát előfordulhat, hogy a legjobb megoldás az volna, ha Gizi és Manci (elnézést kedves Gizella és Mária nevű informatikus olvasóinktól az általánosításért) asztala közé beraknánk egyetlen gépet, amit mindketten használni tudnak. Ez mind helyet, mind költséget, mind pedig felügyeleti időt takaríthat meg számunkra, valamint remekül szórakozhatunk, amikor Gizi egy felelőtlen pillanatában a KDE kiléptető ablakában nem a kilépést, hanem a leállítást választja, miközben Manci éppen egy többoldalas pályázatot készít.
Miguel Freitas például egy egyszerű okból kifolyólag állt neki a második videokártya felélesztésének. Ahogy írja, van egy barátnője és egy számítógépe, de egyszerre csak egyikőjük használhatta a gépet. Mivel egy második kártya és egy második monitor amúgy is jelen volt a gépben (3Dfx-es játékokhoz), gondolta, egy huszárvágással megoldja a gondot.

Alkatrészigény

Már működő rendszerünket ki kell egészítenünk egy monitorral, egy billentyűzettel, egy egérrel és egy videokártyával.
A monitor kapcsán nincs különösebb feltétel, viszont a többiről érdemes pár szót ejteni.

Kétféle billentyűzet létezik: a PS2-es és az USB-csatlakozóval ellátott. Az USB-eszközök nagyon ügyesek, tetszőleges számú, egyetlen gépre akár ugyanolyan típusú USB-eszközöket is rákötethetünk, legyen az akár a sokadik billentyűzet is. Ha csak PS2-es billentyűzeteink vannak, akkor valamiféle turpisságot kell alkalmaznunk. Elméletileg létezik PS2-elosztó (még nem volt szerencsém kipróbálni), és az alaplapok hátulján két PS2-es csatlakozót találunk. Igaz, a második csatlakozó az egérnek van fenntartva, de a Linux az oda csatlakoztatott billentyűzetet is vígan kezeli. Tehát elosztó nélkül akár két PS2-es billentyűzetet is használhatunk, plusz tetszőleges számú USB-csatlakozósat.

Előny	Hátrány
Költségkímélő megoldás	A munkaállomásoknak egymáshoz közel kell lenniük
Egyszerűbb felügyelet	Ha a gép meghibásodik, egyszerre több munkaállomás esik ki
Kevesebb gép, kisebb zaj	Jól megválasztott monitorkártya szükséges

Az egereknél szintén szóba jön az USB, a PS2, de a soros csatlakozós egér is. Ezek közül bármelyik típusúból bármennyit használhatunk, lehetőségeink korlátait a gép csatlakozói szabják meg. Én mind a billentyűzet, mind az egér esetén egy PS2-es és egy darab USB-s eszközt használtam.

A videokártyák beállítása

A videokártya a sarkalatos pont. Igazából azt kell elérnünk, hogy az operációs rendszer elég számú „VGA” típusú eszközt biztosítson. Itt először a kétfejes videokártyák használata jön szóba, mint amilyen például a Matrox G550-esből is akad. Mivel ez a terület most fejlődik, nyugodt szívvel nem mondhatom el, hogy mindegyik kétfejes videokártyát két külön eszközként kezelné a rendszer. A nálam megfordult G400-asnak például csak az első „VGA”-fejét látta a rendszer. A másik kérdés pedig az, hogyha több videokártyát használunk (mert valami csoda folytán rá tudtuk tenni a mancsunkat egy elfogadható minőségű PCI-csatolós videokártyára), akkor a kártyák hajlandóak-e egymás mellett működni.
Fontos tehát, hogy ***ki kell próbálnunk a kártyákat***, csak így bizonyosodhatunk meg, hogy a rendszer megfelelő számú VGA-eszközt biztosít-e számunkra. Az elérhető eszközöket az alábbi paranccsal nézhetjük meg:

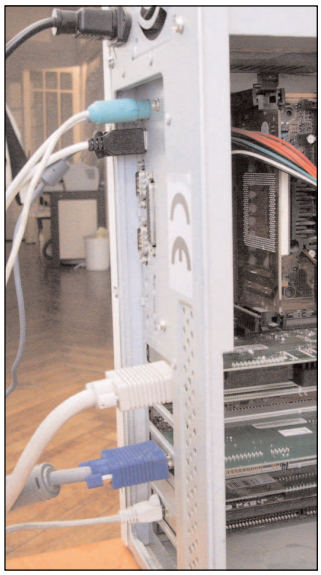
```
root@mancika # less /proc/pci
```

A megjelenő szövegben keressük meg azokat a bekezdéseket, amikben a „VGA” szövegrész szerepel. Itt láthatjuk, hogy milyen nevű eszközök találhatóak a gépben, illetve hogy milyen azono-

sítóval (BusID) rendelkeznek. Az azonosító három számból áll, az első PCI-os kártyáknál 0, AGP-s kártyáknál 1, a második a csatoló sorszáma, míg a harmadik egy ezen belüli sorszám. Példánkban a két videokártya egy PCI-os S3 Virge DX (BusID=0:10:0), valamint egy AGP-s TNT2 (BusID=1:0:0).

A billentyűzetek szétválasztása

Talán ez a lépés tekinthető szakmailag a legnehezebbnek. Ugyanis amíg újra nem írják a rendszermag bemeneti eszközeivel foglalkozó részt, addig sajnos akárhány billentyűzetünk van, alapértelmezésként egyetlen bemeneti eszközként használja a gép. Ez a működés valahol érthető, hiszen a rendszert PS2-es és USB-s billentyűzetről is el szeretnénk tudni indítani. Ahhoz, hogy a rendszermag könnyen kezelhető felületet adjon több billentyűzet egymástól független kezelésére, egy keveset piszkálgatnunk kell. Két lehetőségünk van; a első, hogy az eredeti rendszermagból dolgozunk, a másik, hogy a backstreet ruby magfoltot alkalmazzuk. A folt alkalmazása után néhány függőség megváltozik a magforrásban, például nem lehet ***framebuffer***-eszközöket használni. Ezt a részt röviden átbeszéljük, de aki még nem fordított magot, jobban teszi, ha megkéri az egyik ismerősét, hogy segítsen neki. Tehát kell egy 2.4.20-as magforrás, ezt bontsuk ki, majd igény szerint alkalmazzuk rá a foltot (mind a mag forrása, mind a folt megtalálható a CD-mellékleten a ***Magazin/multix***, illetve a ***Rendszermag*** könyvtárban).



Az itteni beállításokat részletesen ***Aivil Stoss*** „Quick Kernel" oldalán találjuk meg; a lényeg, hogy bekapcsoljuk az ***USB***, ***USB Mouse*** és ***USB HID*** támogatást, és vigyázzunk, hogy az ***USB Keyboard*** ne legyen bekapcsolva (ugyanis abban a pillanatban a rendszermag egy elegáns mozdulattal egyetlen eszközként kezeli a PS2-es és az USB-s billentyűzeteket). Figyeljünk arra is, hogy a ***DUMBCON*** szolgáltatást is használni fogjuk!
Készítsük el a rendszermagot, állítsuk be a `dumbcon=2` indítási kapcsolót is (`append="dumbcon=2"`), majd élesszük fel az új magot. Ha ezzel megvagyunk, a rendszermag csak az elsődleges billentyűt kezeli a konzolon (tehát belépéskor csak a PS2-es billentyűzettel tudunk gépelni, a másik billentyűzet csak csendben ücsörög az asztalon). Azt, hogy a rendszermag megkapja-e a `dumbcon`-kapcsolót, a `dmesg` kimenetében ellenőrizhetjük:

```
root@mancika # dmesg | grep dumbcon

Kernel command line: auto BOOT_IMAGE=2.4.20
➥ro root=301 dumbcon=2
```

Ezek után már csak az a kérdés, hogy léteznek-e a megfelelő eszközök a */dev* alatt. Amennyiben nem, az `mknod` paranccsal hozzuk őket létre (ehhez is találunk részletes leírást a Backstreet Ruby honlapján).

Ellenőrizhetjük, hogy gépünk látja-e a további billentyűzeteket és egereket, ha belenézünk a */proc/bus/input/devices* fájlba. Főpróbaként pedig „hallgassunk bele" az eszközökbe. Amikor az eszközt a `cat` paranccsal kiíratjuk, és elkezdünk gépelni, vagy mozgatjuk az egeret, mindenféle krikszkrakszok jelennek meg a képernyőn. Így megkereshetjük, hogy melyik eszközt milyen néven kezeli a mag. Ha a CTRL+C-vel kilépünk, és szemetes marad a kép, a `reset` paranccsal visszakaphatjuk az eredeti beállításokat.

Egerek

Szerencsére az előző bonyolult folyamat mellékhatásaként az egereket is megtaláltuk – ezeket a rendszer eleve külön kezeli, sőt a létrehozott `mouse0`, `mouse1` stb. eszközökként könnyedén hivatkozhatunk is rájuk. Itt kell elmondanom, hogy míg az X-nek induláskor a `vtn` kapcsolóval adjuk meg, hogy melyik billentyűzetet használja, az egérként használandó eszközt mi magunk adjuk meg. Így ha nem figyelünk, eljuthatunk abba a csuda állapotba is, hogy a két egymástól független X-felület mindkét egérmutatója ugyanazt az egeret követi hűségesen.

XFree86

Ha tehát ott tartunk, hogy a rendszer több VGA-eszközt lát, több billentyűzetünk van, amiket szétválasztottunk, valamint ismerjük egereink eszközneveit, jöhet az `XF86Config-4` nevű pokoljárás. A fájl szerkezetét és használatát a 39. oldalon lévő cikk segítségével megismerhetjük – ez alapján alakítsunk ki egy olyan beállítófájlt, amivel két X-et a megfelelő monitor- és egérbeállításokkal indíthatunk. Figyeljünk arra, hogy ha a két kártya nem ugyanazokat a modulokat támogatja, előfordulhat, hogy két külön beállítófájlt kell használnunk. Ekkor a második beállítófájl használatát az `xf86config` kapcsolóval adhatjuk meg:

```
root@ mancika # startx -- -xf86config
➥/etc/X11/XF86Config-virge
```

Még egy szót kell szólnom az XFree86-tal kapcsolatban. Mindenképpen 4.2.0 vagy nagyobb változatszámú kiszolgálót használjunk, és figyeljünk rá, hogy ez a programcsomag is komoly változásokon megy át. Nem mindegyik változat támogatja ugyanúgy a program több példányának párhuzamos futtatását, főleg a már körbejárt billentyűzetgondok miatt, ezért előfordulhat, hogy trükköznünk kell. Ismétlem, előfordulhat. Ugyanis könnyen lehet, hogy a jelenleg a rendszerünkön lévő XFree86 is tökéletesen képes ellátni a feladatot (ez függ a videokártyáktól, sőt az alaplaptól is).
Ha mégis trükközünk, akkor választhatunk a régebbi, de többek által kipróbált, illetve az új lehetőség között. A sokak által járt út az, amikor valahogy a rendszerünkre varázsoljuk a 4.2.0-s változatot (sajnos az xfree86.org alatt már nem találjuk meg, de még több fájlkiszolgálón előfordul, valamint a magazin lemezmellékletére is felkerült), majd Aivil Stoss „Quick XFree" oldalán leírt módon használjuk a foltozott futtatható fájlt:

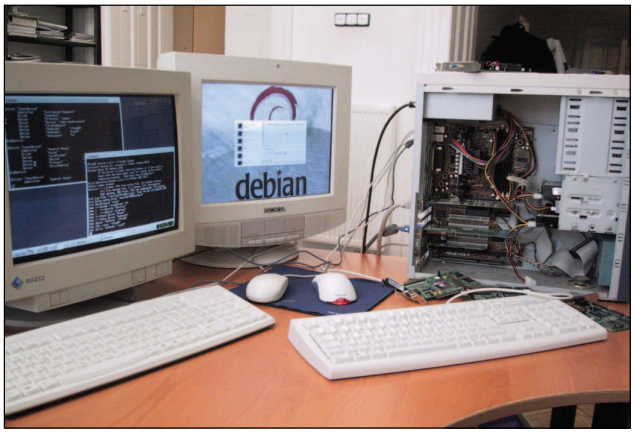
```
# cd /usr/X11R6/bin
# mv X X.old
# mv XFree86 XFree86.old
# cp /cdrom/XFree86/XFree86-patched .
# chmod +x XFree86-patched
# ln -s XFree86 XFree86-patched
# ln -s X XFree86-patched
```

Ennek a módszernek viszont nagyon komoly hátulütője, hogy

vigyáznunk kell, nehogy valamelyik csomagfrissítéskor a rendszer felülírja. Mivel a számunkra szükséges kódrészlet előbb-utóbb az XFree86-ba bekerül és üzembiztossá válik (vagy talán már vált is), a másik lehetőségünk az, hogy a legfrissebb 4.3-as változatot használjuk.

Szerencsejáték

Igen, most jön a lutri. Ki kell sakkoznunk, hogy az általunk használt monitorkártyák egymás mellett működnek-e. Ha valakinek nagyon nagy szerencséje van, akkor egyszerűen elindítja a két grafikus felületet, és már vígan használja is. Ezeknek az olvasóimnak szívből ajánlom, hogy próbálkozzanak meg a lottóval is, mert jó esélyük van egy ötös találatra.



Hogy melyik billentyűzetet használjuk, azt az X-nek a fentebb említett `vtn` kapcsolóval adhatjuk meg. Ha rendszermagunk a `dumbcon=2` értékkel indult, akkor a `vt7` az első, a `vt9` a második, a `vt17` pedig a harmadik billentyűzetet használja (az egeret a ***layout*** részben megadott egér eszközneve alapján választja ki a rendszer). Próbaként indítsuk el az első grafikus felületet, majd egy `xterm`-ből indítsuk el a másodikat:

```
root@mancika # startx -- :0 vt7 -layout egyes
...
xterm # startx -- :1 vt9 -layout kettes
```

Nézzük, milyen eredménnyel járhat a próba:

- Működik – minden gond megoldva.
- Fagyás – a második X elindításakor a gép lefagy, semelyik billentyűzet, semelyik egér nem válaszol.
- Az első várakozik – az első X-felület nem változik, de amikor a másodikat bezárjuk, az első újra feléled, a közben begépeltek megjelennek stb.
- Tetszhalott – maga az első X fut, ha parancsot adunk ki, végrehajtja (lehet valami könnyebben ellenőrizhetőt is használni, nem csak a `reboot`-ot), de a monitoron nem frissül a kép.
- Bizonytalan működés – egy idő után vagy például üzemmódváltás (ha több felbontást állítottunk be, akkor a CTRL+ALT+<JOBB OLDALI +> kombinációval váltogathatunk) esetén a gép fagy.

Mitől függhet az eredmény? Elsősorban, ahogy azt már írtam, a használt videokártyáktól. De még itt sem mondhatjuk, hogy az egyik kártyapárosítás biztos jó, míg a másik biztosan nem. ***Aivil Stoss*** honlapján egy külön lapon (***Video card compatibility)*** igyekszik összegyűjteni azokat a kártyákat, amikről már biztos adatokkal rendelkeznek. De például az én próba gépemben az S3Virge a Voodoo3 mellett várakozott, TNT2-es mellett viszont fagyott. Érdekes, hogy Aivil Stoss éppen egy TNT2-est használ egy Voodoo3-assal, nekem pedig nem akarnak egymás mellett futni. A másik fontos tényező, hogy milyen X-et használunk: egy friss 4.2-eset, a foltozott 4.2.0-s változatot vagy az új 4.3-ast? Emellett figyeljünk rá, hogy nVidia-lapkás monitorkártya esetén szinte mindig jobban járunk, ha letöltjük és telepítjük az új nVidia magmodult (lásd a 40. oldalon lévő tippet).
Fontos kérdés, hogy melyik kártyával indul a rendszer. Úgy tapasztaltam, hogy jobb a „gyengébb" kártyát alapértelmezettnek választani. Ha egy PCI és egy AGP-csatolós kártyánk van, akkor a BIOS-ban választhatjuk ki, melyik legyen az alapértelmezett, két PCI-csatolós esetén pedig a kártyák sorrendje dönt. És hogy ez szép legyen, az sem mindegy, hogy melyik kártyán indul el először az X-felület. Ennek – tapasztalatom szerint – a legtöbbször a rendszer alapértelmezett kártyájával kell azonosnak lennie (tehát ugyanazon a kártyán indítsuk a vt7-en futó X-et, amelyiken a rendszer indult).
Azt már csak csendben említem meg, hogy nem mindig mindegy, hogy a második X-et az első X alól vagy egy karakteres konzolról indítjuk-e (pontosabban, hogy a második X indulásakor az első videokártya milyen üzemmódban van), ugyanis sok kártya esetén az üzemmódváltás következetesen rendszerfagyást okoz. Fura módon ez nem mindig igaz, mert például a nálam lévő S3 Trio 64 V2 esetében pont egy üzemmódváltással tudtam tetszhalott állapotából felélesztení.
Az örömhír az, hogy a kétfejes videokártyák (amelyeket két külön videokártyának lát a rendszer, ilyen például a Matrox G550) esetén állítólag hibátlan a működés. Sajnos, nem volt lehetőségem ilyen kétfejes kártyát kipróbálni.

Zárszó

Mint látjuk, jelen pillanatban nem éppen leányálom egy ilyen több-munkaállomásos gép összerakása; szerencsére – mint minden terület – a kialakításhoz szükséges elemek is folyamatosan fejlődnek. Remélhetőleg az új rendszermag és az új X már az összes szükséges beállítást támogatja, és hamarosan az egész cikk helyett csupán egyetlen bekezdést kell majd írnunk: „Végy egy kétfejes monitorkártyát, dugj a gépbe még egy billentyűt és még egy egeret, és indíts el egy második grafikus felületet a `startx -- :1 vt9 -layout second` paranccsal." Amíg ez nem így zajlik, marad a „betyárkodás".

Szy György (Szy.Gyorgy@linuxvilag.hu)
Már hosszú ideje kedvenc szórakozása, hogy minél kevesebb géppel tudja megoldani a szerkesztőség „gépigényét".

KAPCSOLÓDÓ CÍMEK

Backstreet Ruby – Aivil Stoss tolmácsolásában
➲ http://startx.times.lv/

Miguel Freitas megoldása:
➲ http://cambuca.ldhs.cetuc.puc-rio.br/multiuser/

Az XFree86 4.2.0-s teljes változat egyik lelőhelye
➲ http://www.go.dlr.de/fresh/linux/XFree86/4.2.0/binaries/Linux-ix86-glibc22/

XF86Config-4

Az XFree86 grafikus felület alapvető működését meghatározó beállításokat a */etc/X11/XF86Config-4* nevű fájl tartalmazza. Tekintsük most át a fájl felépítését és részeit! Az egyszerű szöveges állomány szakaszokra van bontva, minden szakasz a következő szerkezetet követi:

```
Section "szakaszneve"
       Kapcsol       ØrtØk(ek)
       . . .
EndSection
```

A rendszer a rendszergazdaként kiadott `X -configure` parancs hatására */root/XF86Config-new* néven egy mintafájlt hoz létre, amelyben feltünteti az általa felismert eszközöket. Ez a fájl jó alapot jelent azXFree86 beállításához. Nézzük most végig, milyen szakaszok találhatók benne.

Files

A rendszer által használt fájlok elérési útjait adhatjuk meg, a három legfontosabb beállítás az *RgbPath*, ami a színkezeléshez szükséges fájlok helyét mutatja; a *ModulePath*, ami a rendszer bővítményeinek helyét adja meg; valamint tetszőleges számú *FontPath*, amelyekben megadhatjuk az összes használni kívánt betűkészlet elérési útvonalát. A betűkészletek kapcsán egy dolgot kell kiemelnünk: egy adott típusú betűkészletet a rendszer csak akkor használ, ha a hozzá tartozó modult is betöltjük; ha például az Adobe által kifejlesztett type1-típusú készleteket használni akarjuk, a Module részben szerepelnie kell a `type1` modulnak. Ugyanígy a TrueType betűkészleteket vagy a `freetype`, vagy `ttf` modul kezeli.
Másik lehetőségünk egy betűkiszolgáló telepítése (`xfs`, `xfs-xtt`, `xfstt` stb.), a betűkészleteket a rendszer vagy helyi foglalaton, vagy TCP-kapcsolaton keresztül éri el (tehát így oldható meg, hogy a hálózat egyik gépére telepített betűkészleteket a hálózat összes gépe használni tudja):

```
         FontPath      "unix/:7100"
# a 7100-es foglalaton csatlakozik a helyi
# gØpre
         FontPath      "tcp/mancika:7100"
# Mancika gØp 7100-as kapujÆn elØrheti
# kiszolgÆl t hasznÆlja
```

Module

A fájlban egyetlen *Module* nevű szakasz lehet, ami az X által betöltendő programmodulok neveit tartalmazza – ide tartoznak például a különböző betűkezelő modulok (`speedo`, `type1`, `freetype`, `xtt`, `...`), valamint a megjelenítés gyorsítását segítő modulok (`dri`, `glx`, `...`). A modulok a */usr/X11R6/lib/modules/* könyvtárban találhatók, az előbbiek a *fonts*, az utóbbiak az *extensions* alkönyvtárban. Ebben a könyvtárban érdemes körbenézni, itt találkozhatunk az összes videokártya-illesztő modullal is (*drivers* alkönyvtár), vagy például a rendszer által használt beviteli eszközök illesztőivel (*input* alkönyvtár). E két utóbbi típusra más szakaszokban hivatkozunk majd.

ServerFlags

Ebben a szakaszban további beállításokat adhatunk meg, ha például a később említésre kerülő xinerama kiterjesztést állandóan használni szeretnénk, az alábbi módon állíthatjuk be:

```
Section "ServerFlags"
    Option "Xinerama" "true"
EndSection
```

Többször előforduló szakaszok

A továbbiakban olyan szakaszok következnek, amelyekből több is szerepelhet a beállításfájlban. Ezeket a rendszer típusuk és típusonként egyedi azonosítójuk (`Identifier` kapcsoló) alapján különbözteti meg. Az azonosítónak bármilyen szöveges érték adható, mégis azt ajánlom, hogy ragaszkodjunk a megszokott egyszavas nevekhez, egy sorszámmal egészítve ki a szó végét olyan eszközök esetében, amelyekből több lehet (mouse0, mouse1 stb.).

InputDevice

Ahogy neve is mutatja, itt beviteli eszközöket adhatunk meg, külön szakaszt hozva létre minden egyes beviteli eszközhöz. A rendszer alapértelmezésként egy billentyűzetet és egy egeret állít be. Jelenleg nincs szükség több billentyűzet beállítására, azt a kiszolgáló indításakor a `vtn` kapcsoló határozza meg. Hasonlóan külön-külön szakaszt kell létrehoznunk például a digitábláknak vagy az érintőképernyőknek is.
A billentyűzetnél megadhatjuk, hogy milyen szabályokat akarunk használni (a különböző vezérlőbillentyűk jelentése stb.), milyen elrendezésű billentyűzetünk van (hány gomb található rajta), illetve melyik nyelvnek megfelelő kiosztás legyen az alapértelmezett. A kiosztást később a rendszer futása közben a `setxkbmap` paranccsal bármikor megváltoztathatjuk.
Az egér esetében meg kell adnunk, hogy melyik rendszereszközön keresztül használjuk (soros egérnél ez az egyik /dev/ttyS*nn*, PS/2-esnél a */dev/psaux* az új bemeneti réteget használó egereknél, mint amilyenek az USB-egerek, az egyik a /dev/input/mouse*n)*, valamint hogy az egér milyen protokollt használ (ez általában PS/2, görgős egérnél IMPS/2). Egy további érdekes lehetőség, hogy ha többgörgős vagy kiegészítő gombokkal is ellátott egerünk van, a rendszerrel tudathatjuk, hány különböző gomb kezelésére készüljön fel (egy görgő két gombot jelent), illetve hogy ezek melyike hogyan legyen tengelyekhez rendelve (lásd a `ZAxisMapping` kapcsolót). Természetesen minden egerünkhöz külön szakaszt kell létrehozzunk.

```
Section "InputDevice"
         Identifier "Keyboard0"
         Driver     "keyboard"
         Option     "XkbRules"     "xfree86"
         Option     "XkbModel"     "pc105"
         Option     "XkbLayout"    "hu"
EndSection

Section "InputDevice"
```

```
        Identifier "Mouse0"
        Driver     "mouse"
        Option     "Buttons"      "9"
        Option     "Protocol"     "IMPS/2"
        Option     "Device"       "/dev/psaux"
        Option     "ZAxisMapping" "4 5 8 9"
EndSection

Section "InputDevice"
        Identifier "Mouse1"
        Driver     "mouse"
        Option     "Buttons"   "5"
        Option     "Protocol"  "IMPS/2"
        Option     "Device"
"/dev/input/mouse1"
        Option     "ZAxisMapping"  "4 5"
EndSection
```

Monitor

Ugyan a rendszer az újabb monitoroknál elméletileg magától meg tudja állapítani, hogy azok milyen tudással bírnak (a legnagyobb felbontás, különleges képességek), egyelőre mégis kézzel állíthatjuk be az adatokat. Ezeket az a legjobb, ha a monitor leírásában megadott adatokhoz igazítjuk. A rendszer ez alapján fogja megtalálni a legjobb, a monitorral még használható üzemmódot. Gyakran előfordul, hogy a grafikus felület csak nem akar magasabb frekvencián működni, hiába volna képes rá a monitor és a videokártya is. Nos, a rendszer gyakran az itt található `HorizSync` és `VertRefresh` értékek miatt nem engedi a magasabb képfrissítés használatát. A `VendorName` és `ModelName` értéke lényegtelen, egyszerű szöveges leírások.

```
Section "Monitor"
        Identifier   "Monitor0"
        HorizSync    30-95
        VertRefresh  40-75
        Option       "DPMS"
        VendorName   "Monitor Vendor"
        ModelName    "Monitor Model"
EndSection
```

Amennyiben több azonos tudású monitorunk van, elég egyetlen ***Monitor*** szakaszt létrehozni, hiszen semmilyen eszközfüggő beállítást nem találunk – innen a rendszer csak az adatokat használja.

Device

Ez az egyik legfontosabb szakasz, videokártyánk adatait állíthatjuk itt be. A két alapadat a használandó vezérlőmodul (***Driver***), valamint a sínazonosító (***BusID***). Amennyiben csak egy kártya van a gépünkben, illetve a használni kívánt vezérlőmodul a gépben található kártyák közül csak az egyiket hajlandó vezérelni, a BusID el is hagyható – ennek ellenére a használata javasolt. A gépben lévő kártyák azonosítója könnyen kikövetkeztethető. Nézzünk bele a */proc/pci* fájlba, és keressük meg a videokártya bejegyzését. Valami ilyesmit láthatunk:

```
Bus  1, device   0, function  0:
  VGA compatible controller: nVidia
  Corporation NV11 [GeForce2 MX] (rev 161).
    IRQ 16.
    Master Capable.  Latency=248.
    Min Gnt=5.Max Lat=1.
    Non-prefetchable 32 bit memory at
    0xde000000 [0xdeffffff].
    Prefetchable 32 bit memory at
    0xd0000000 [0xd7ffffff].
```

Ami érdekes, az az első sor. Bár ez egy AGP-s videokártya, a rendszer ezt az eszközt is „szabványos PCI-rendszeren” keresztül kezeli. A példában látható kártya azonosítója (a `PCI:<bus>:<device>:<function>` összeállítás alapján) `PCI:1:0:0`. Gépemben egy Geforce típusú kártya található, így két vezérlőt is használhatok, az egyik az XFree86-hoz adott `nv` vezérlő, a másik az nVidia oldaláról letölthető nVidia-vezérlő (telepítést, lásd a dobozban). Ha gyorsítás szükséges, mindenképpen javaslom az nVidia használatát.
Egy további jó szolgáltatása az `X -configure` parancsnak, hogy a mintafájlban felsorolja az összes, az adott vezérlővel használható kapcsolót, így nem nekünk kell kimazsoláznunk, hogy melyiket hogyan is hívják. Ezek a kapcsolók általában finomhangolásra használhatók.

```
Section "Device"
        #Option      "NvAgp"         "1"
. . .
        Identifier   "Card0"
        Driver       "nvidia"
        VendorName   "NVidia"
        BoardName    "GeForce2 MX"
        BusID        "PCI:1:0:0"
EndSection
```

Az nVidia saját verzérlője

Azt hiszem, nem sokat tévedek, ha azt állítom, hogy manapság a leggyakrabban valamilyen nVidia-lapkás videokártyával találkozhatunk gépeinkben. Ide tartoznak a régebbi Riva, TNT és TNT2-es kártyák vagy az újabb Geforce-család. Sőt némelyik alaplapra már ráépítve találkozhatunk az nForce lapkával is.
Hogyan tudjuk kiaknázni ezek egyedi képességeit? Az XFree86 ugyan alapállapotában rendelkezik egy `nv` vezérlővel, de ennek a tudása korlátozott. Szerencsére az nVidia is készít egy vezérlőt, újonnan – vajh, miért? – `nvidia` névre keresztelve. Mivel a gyártó a vezérlőkódot zárt forrással készíti, az nem képezi részét az XFree86-csomagnak. Ennek vannak jó és rossz mellékhatásai. A jó, hogy a gyártó belső emberei készítik a kódot, ami ezáltal tényleg igyekszik kiaknázni a lapkák tudását. Olyannyira, hogy az nvidia.com alatt található fórumon több Unreal-rajongó elújságolta, hogy a gépén a windowsos változatnál átlag tíz százalékkal gyorsabb megjelenítést kapott.
A rossz hír, hogy a vezérlő nagy része magmodulként fut. Méghozzá nem is kis része! A modul majdnem másfél megabájt méretű! Igaz, leegyszerűsödik az életünk, ugyanis egyetlen vezérlő kezel minden újabb típusú (TNT utáni) nVidia-lapkát. Akkor ajánlott tehát a vezérlő telepítése, ha gépedben nVidia-lapkás videokártya van, és játékra is használni szoktad a gépet. A telepítés nem túl bonyolult, a szokásos letölt–kicsomagol–`make install` hármas.
A telepítés után még két feladat vár ránk, ha a telepítő ezeket magától nem végzi el. Az első, az `nvidia.o` modul betöltése a rendszermagba (`insmod nvidia.o`), a másik pedig az *XF86Config-4* fájl frissítése (az `nv` vezérlő helyett `nvidia` vezérlő, továbbá ha még nem szerepel a modulok között, akkor add hozzá a `GLX` modult is).
Jó játékot!
➲ http://www.nvidia.com/content/drivers/drivers.asp

Screen

Mivel több videokártyánk és monitorunk lehet, sőt ezeket eltérő módon is használhatjuk, az X számára egy-egy `Screen` szakaszban állíthatjuk be, hogy az adott név alatt pontosan melyik videokártyához kötött melyik monitort szeretnénk használni, s azt milyen színmélységgel és milyen felbontásban. A későbbiekben ezekre mint képernyőkre hivatkozhatunk. Természetesen mindegyik `Screen` szakaszban több színmélységet is megadhatunk, mindegyikhez külön-külön meghatározva (egy-egy `Display` alszakaszban), hogy milyen felbontásban kívánjuk használni. Általánosan azonban csak egyet használunk, ami a legtöbb esetben a 24-bites színmélység. Amennyiben az adott színmélységhez több felbontást is megadunk, a grafikus felület használata közben a CTRL+ALT nyomva tartása mellett a számbillentyűk melletti plusz- és mínuszjelekkel váltogathatunk közöttük.

```
Section "Screen"
        Identifier     "Screen0"
        Device         "Card0"
        Monitor        "Monitor0"
        DefaultDepth 24
        SubSection     "Display"
           Depth       16
           Modes       "1024x768" "800x600"
        EndSubSection
        SubSection "Display"
           Depth       24
           Modes       "1024x768" "800x600"
        EndSubSection
EndSection
```

Mindegyik használni kíván felülethez (kártyába, monitor, üzemmód) külön `Screen` szakaszt kell létrehozzunk.

ServerLayout

És végül az utolsó szint, ahol az egészet egybepakoljuk, és egyetlen névként hivatkozunk az elrendezésre, a `Layout` szakasz. Az elrendezés több képernyőt is összefoghat, ezt a megoldást elsősorban a Xinerama kiegészítés használja – ebben az esetben egyetlen „összetett” képernyőként kezeli a több képernyőből álló rendszert. Az ilyen többmonitoros rendszereknek például a grafikusok látják hasznát. Aki már próbált térszerkesztővel és képszerkesztővel egyszerre dolgozni, miközben folyamatosan látni akarta az előnézetet, az igazán értékelni tudja több képernyő együttes használatának a lehetőségét.
De más oka is lehet a több kiépítés használatának. Például hordozható gépünk van, és amíg a gép beépített megjelenítője csak a 800×600-as felbontást ismeri, addig az irodában asztali monitorunkon inkább az 1280×1024-es felbontás örömeit szeretnénk élvezni. Annyi a dolgunk csupán, hogy két különböző elrendezést hozunk létre, és a grafikus felületet mindig a megfelelő kapcsolóval indítjuk. Az alábbi példában mindhárom lehetőséget igyekeztem szemléltetni:

```
Section "ServerLayout"
        Identifier     "alap"
        Screen      0  "Screen0" 0 0
        InputDevice    "Mouse0" "CorePointer"
        InputDevice    "Keyboard0"
                       ➥"CoreKeyboard"
EndSection

Section "ServerLayout"
        Identifier     "nagy"
        Screen      0  "Screen1" 0 0
        InputDevice    "Mouse0" "CorePointer"
        InputDevice    "Keyboard0"
                       ➥"CoreKeyboard"
EndSection

Section "ServerLayout"
        Identifier     "ketkepernyo"
        Screen         "Screen0" 0 0
        Screen         "Screen2" Rightof
                       ➥"Screen0"
        InputDevice    "Mouse0" "CorePointer"
        InputDevice    "Keyboard0"
                       ➥"CoreKeyboard"
EndSection
```

Ekkor (természetesen a megfelelő Screen*n* és Card*n* szakaszok létrehozása után) a felületet több kiépítéssel indíthatjuk:

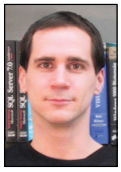
```
root@mancika # startx -- -layout alap
root@mancika # startx -- -layout nagy
root@mancika # startx -- -layout ketkepernyo
               ➥+xinerama
```

Hogyan indítsuk a felületet?

Amíg a beállításokkal szöszölünk, érdemes leállítani a grafikus bejelentkezéskezelőt (ha éppen használunk ilyet), `kdm` esetén például a `/etc/init.d/kdm stop` parancs segítségével. A grafikus felületet három paranccsal közvetlenül is indítani tudjuk:

- Az `X kapcsol k` csak a nyers grafikus felületet indítja el.
- Az `xinint [alapprogram] -- kapcsol k` elindítja a felületet, majd az ***alapprogramot***. Ez a program – ha mást nem adunk meg – egy egyszerű `xterm`. A program bezárása a grafikus felület bezárását vonja maga után.
- A `startx [ gyfØl] -- kapcsol k` paranccsal, ami valójában az `xinit`-et hívja meg.

A másik lehetőségünk, hogy a grafikus felülete(ke)t a bejelentkezéskezelővel indíttatjuk. Ebben az esetben vigyáznunk kell rá, hogy nem mindegyik támogatja ugyanolyan mértékben a többképernyős elrendezést, illetve a Xinerama kiterjesztést. A kde3-hoz tartozó `kdm` használata esetén az elindítandó grafikus felület pontos megadását a ***/etc/kde3/kdm/Xservers*** fájlban tehetjük meg.

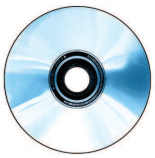


Szy György (Szy.Gyorgy@linuxvilag.hu)
Már hosszú ideje kedvenc szórakozása, hogy minél kevesebb géppel tudja megoldani a szerkesztőség „gépigényét”.

KAPCSOLÓDÓ CÍMEK

A Xinerama használatának leírása
➲ http://www.tldp.org/HOWTO/Xinerama-HOWTO/

Az XF86Config-4 fájlhoz tartozó súgóoldal is sok hasznos adatot tartalmaz: man XF86Config-4



Az USB soros illesztőprogram-réteg

Az USB-soros réteg ismertetése, valamint útmutató az eszközök illesztéséhez.

Legutóbbi írásomban (a Linuxvilág 2002. decemberi számában) a 2.5-ös (és remélhetően hamarosan a 2.6-os) rendszermagfa soros rétegét ismertettem. Futólag megemlítettem, hogy a rendszermag USB-soros illesztőprogram-rétege segítségével könnyebben dolgozhatunk az ilyen típusú illesztőprogramokkal. Most elérkezett az ideje annak, hogy megismerkedjünk ezzel a réteggel.

Az USB-soros réteg története

Réges-rég (legalábbis a rendszermag fejlesztését nézve) készült egy USB-soros illesztőprogram, ami bekerült a rendszermagfába. Sajnos csak egyetlen eszköztípussal működött, SMP (többprocesszoros) gépeken pedig egyáltalán nem lehetett használni. Mivel nem volt szabványos USB-soros protokoll, minden eszköz egyedi, az egyes gyártók által kigondolt protokollt alkalmazott. Miért nincs szabványos protokoll…? Hosszú és csúnya történet ez, akit érdekel, az a ***linux-usb-devel*** levelezőlista levéltárában megtalálja a választ. Hamarosan egy második USB-soros eszköztípus támogatása is megvalósult a már említett illesztőprogramon belül, lefoglalva a fő- és alváltozatszámok egy részét. Idővel egyre több és több eszközt adtak hozzá az illesztőprogramhoz, áttekinthetetlen maszlagot hozva létre. Ekkor ***Peter Berger*** és ***Al Borchers*** segítségével az illesztőprogram eredeti szerzője az egész alrendszert újraírta, létrehozva ezzel az USB-soros réteget. Munkájuknak köszönhetően csekély mennyiségű kódot írva különféle USB-soros illesztőprogramokat lehet készíteni, és ezek ugyanazt a fő- és alverziószám-tartományt használják. Az egyes illesztőprogramok elől részben sikerült elfedni a tty- és az USB-réteg bonyolult működését. Az illesztőprogramok különálló modulokba is fordíthatók, és szükség szerint tölthetők be.
A 2.5-ös rendszermag fejlesztésekor a soros réteget azért készítették, hogy – szükségtelenné téve a tty-réteg közvetlen kezelését – megkönnyítsék a soros kapukhoz szánt illesztőprogramok fejlesztését. Reméljük, hogy a jövőben az USB- és a soros réteg egybeolvad. A kódok karbantartói is ezt szeretnék, csak éppen nincs idejük a megvalósításra. (Természetesen örömmel fogadnak minden erre irányuló foltot, ha valaki nem találna neki tetsző tervezetet.)
Ebben az írásban az USB-soros réteggel kapcsolatos alapvető tudnivalókat tárgyalom, szó lesz az illesztőprogramok bejegyzéséről és a bejegyzések törléséről, illetve a működésükhöz szükséges fontosabb adatszerkezetek kezeléséről.

Az USB-soros illesztőprogramok bejegyzése és törlése

A cikkben szereplő kódrészletek és példák a 2.5, illetve 2.6-os rendszermagfából származnak. A 2.4-es és a 2.2-es fák is támogatják az USB-soros illesztőprogramokat, de ezeknél az illesztési felület néhány részletben eltérő. Az egyszerűség kedvéért most csak egy rendszermagfával foglalkozunk. Ha egy – 2.5-ös rendszermag alatt már futó – USB-soros illesztőprogramot nem tudsz átültetni a régebbi rendszermag alá, vedd fel velem a kapcsolatot. Ahhoz, hogy egy USB-soros illesztőprogram bejegyzése a rendszermagnál megtörténjen, két dolgot kell tennie: az USB-soros magnál és az USB-magnál egyaránt be kell jegyeznie magát. Az USB-soros magnál végzett bejegyzés teszi lehetővé az illesztőprogram meghívását abban az esetben, ha az USB alrendszer új eszközt talál, az USB-maggal pedig azt kell közölni, hogy az illesztőprogram milyen típusú eszközöket tud kezelni. Amikor az USB-magnál akarod bejegyeztetni az illesztőprogramot, csupán az általa kezelt USB-eszközöket kell felsorolnod, hagyományos USB eszközazonosító formátumban:

```
static struct usb_device_id id_table [] = {
        {USB_DEVICE(TERM K_AZONOS˝T JA,
        ➥ESZK Z_AZONOS˝T JA)},
        {}      /* A bejegyzØs lezÆrÆsa */
};
MODULE_DEVICE_TABLE (usb, id_table);
```

A táblázat alapján az USB-mag tudni fogja, hogy az illesztőprogram milyen eszközöket tud kezelni; a felhasználói területen futó, az üzem közbeni eszközcserékért felelős kódrész pedig nyilván tudja tartani a használatban lévő eszközfajtákat. A Linuxvilág 2002. májusi számában megjelent „A PCI Hot Plug eszközillesztő fájlrendszer működése” című írásomban további tudnivalókat is találsz a táblázatról, illetve arról, hogy az üzem közbeni eszközcseréket támogató kódrész hogyan használja.
Ezután egy egyszerű USB illesztőprogram-adatszerkezet jön létre az azonosító felhasználásával:

```
static struct usb_driver tiny_driver = {
    .name =         "tiny",
    .probe =        usb_serial_probe,
    .disconnect =   usb_serial_disconnect,
    .id_table =     id_table,
};
```

A `.probe` és a `.disconnect` mezőknek az USB-soros mag függvényeire kell mutatniuk, ezeket a feladatokat ugyanis neki, és nem a te illesztőprogramodnak kell kezelnie.
Az USB-magnál a bejegyzés egy egyszerű hívással megtörténik:

```
usb_register (&tiny_driver);
```

A következő hívás révén értesítjük az USB-soros réteget az új illesztőprogramról:

```
usb_serial_register (&tiny_device);
```

A függvény egy `usb_serial_driver_type` adatszerkezetre irányuló mutatót ad vissza, amiről később még bővebben is szó esik. Ha törölni szeretnénk az illesztőprogram bejegyzését, ugyanezeket a lépéseket kell végrehajtanunk, csak fordított sorrendben. Először az USB-soros magnál végezzük el a törlést:

```
usb_serial_unregister (&tiny_device);
```

Ezután kerül sor az USB-magra:

```
usb_unregister (&tiny_driver);
```

Az usb_serial_device_type adatszerkezet

Az USB-soros magnál végzett bejegyzéshez fel kell tölteni az `usb_serial_device_type` adatszerkezetet. Az adatszerkezet meghatározása a ***drivers/usb/serial/usb-serial.h*** fájlban található:

```
struct usb_serial_device_type {
  struct module *owner;
  char *name;
  const struct usb_device_id *id_table;
  char num_interrupt_in;
  char num_bulk_in;
  char num_bulk_out;
  char num_ports;

  struct list_head driver_list;

  int (*probe) (struct usb_serial *serial);
  int (*attach) (struct usb_serial *serial);
  int (*calc_num_ports) (struct usb_serial
                         ➥*serial);
  void (*shutdown) (struct usb_serial *serial);

  int  (*open) (struct usb_serial_port *port,
                ➥struct file * filp);
  void (*close) (struct usb_serial_port *port,
                 ➥struct file * filp);
  int  (*write) (struct usb_serial_port *port,
                 ➥int from_user,
                 ➥const unsigned char *buf,
                 ➥int count);
  int  (*write_room) (struct usb_serial_port
                      ➥*port);
  int  (*ioctl) (struct usb_serial_port *port,
                 ➥struct file * file,
                 ➥unsigned int cmd,
                 ➥unsigned long arg);
  void (*set_termios) (struct usb_serial_port
                       ➥*port, struct termios
                       ➥*old);
  void (*break_ctl) (struct usb_serial_port
                     ➥*port, int
                     ➥break_state);
  int  (*chars_in_buffer)
         (struct usb_serial_port *port);
  void (*throttle) (struct usb_serial_port
                    ➥*port);
  void (*unthrottle) (struct usb_serial_port
                      ➥*port);

  void (*read_int_callback)(struct urb *urb);
  void (*read_bulk_callback)(struct urb *urb);
  void (*write_bulk_callback)(struct urb *urb);
};
```

Meglehetősen méretes adatszerkezet, de még mindig kisebb, mint a tty-rétegé vagy a soros réteg adatszerkezeteinek a halmaza – ezekkel lehetne ugyanis kiváltani az USB-soros réteg használatát. Lássuk, hogy a fenti mezők közül mikre van szükség, és pontosan mi is a szerepük. Az `owner` (tulajdonos) mező egy mutató az eszközt birtokló modulra. A `THIS_MODULE` makróra kell mutatnia. Ilyenkor a modulra mutató hivatkozások számlálóját az USB-soros mag kezeli, ami jóval biztonságosabb módszer annál, mintha magad próbálkoznál vele.

A `name` (név) mező az eszköz leírását tartalmazó karakterláncra mutat. Az eszköz csatlakoztatásakor és leválasztásakor ez a karakterlánc jelenik meg a `syslog` üzenetekben. A */proc/tty/driver/usb-serial* fájlban is használja a rendszer, így állapítható meg, hogy egy-egy eszköz melyik kapura csatlakozik.

Az `id_table` mező egy `usb_device_id` adatszerkezetekből álló listára mutat, ami az adatszerkezet által támogatott eszközöket adja meg. Ez a mező akár azonos is lehet az USB-magnak átadott mutatóval. Ha az illesztőprogramodnak különböző eszköztípusokkal különböző műveleteket kell elvégeznie, akkor természetesen különböző adatszerkezeteket is használhatsz az eszközök leírására. Az utóbbira jó példa a Keyspan illesztőprogram, ami az összes Keyspan USB-soros eszközt kezeli, és eltérő eszközökhöz más és más függvényeket kell meghívnia.

A `num_interrupt_in` mező az eszköz végpont-megszakításainak várható száma. A végpont fogalmát az USB esetében a szabvány vezeti be és írja körül (➲ http://www.usb.org). Ha nem akarsz foglalkozni vele, hogy az USB soros mag ellenőrzi-e ezt az értéket (összeveti-e a felismert eszközökkel), akkor használd az *usb-serial.h* fájlban megadott `NUM_DONT_CARE` makrót.

A `num_bulk_in` és `num_bulk_out` mezők az eszköz tömeges (bulk) adatmozgatásra szolgáló ki- és bemeneti végpontjait adják meg. Itt is használhatod a `NUM_DONT_CARE` makrót, ha valamilyen okból nem akarsz különösebb figyelmet szentelni ennek az értéknek.

A `num_ports` mező adja meg az eszköz különféle kapuinak a számát. Egyetlen USB-soros eszköz is számos különböző fizikai soros kapuval rendelkezhet.

A `driver_list` mezőt az USB-soros mag használja a nála bejegyzett illesztőprogramok nyilvántartására, maguk az illesztőprogramok tehát nem nyúlhatnak hozzá.

Az adatszerkezet egyéb mezői elhagyható függvényekre irányuló mutatók. Ha valamelyik mező értéke nincs megadva, a rendszer az általános USB-soros illesztőprogram megfelelő eljárását fogja meghívni. Így – ha az elvárt működés jelentős részben egyezik az általános illesztőprograméval – gyakorlatilag csekély programozási munkával készíthető illesztőprogram. Ellenkező esetben természetesen ezeket a függvényeket is meg kell írni. A függvénymutatók három csoportba sorolhatók: USB-életciklus mutatók, tty-életciklus mutatók és URB- (USB request block, azaz USB-kérésblokk) visszahívó (callback) mutatók.

Az USB-életciklus függvénymutatói a következők: `probe`,

Az általános USB-soros illesztőprogram

Az általános USB-soros illesztőprogram a `write()` hívás és az eszköz tömeges adatmozgatást szolgáló kimenő végpontja között továbbítja az adatokat. Ugyancsak közvetít a bejövő tömeges adatmozgató végpont és a tty-réteg átmeneti tárai között, így a felhasználó `read()` hívásokkal juthat hozzá az adatokhoz. Ha a te eszközöd így működik, akkor használd az általános illesztőprogramkódot. Egyéb esetben az illesztőprogramodnak kell megadnia az `usb_serial_device_type` adatszerkezetben található függvénymutatókat.

calc_num_ports, attach és shutdown. Meghívásukra különféle időpontokban, az USB-eszköz üzembe helyezésekor és leállításakor kerül sor. A probe függvényre akkor van szükség, amikor egy az id_table listában szereplő eszközök valamelyikének megfelelő eszközt csatlakoztatunk a rendszerhez. Meghívása még az előtt történik meg, hogy az USB-soros mag befejezné az eszköz üzembe helyezését (inicializálását). A szükséges belső programnak (firmware) az eszközre való áttöltésére használható. Ekkor nyílik mód egyéb, az eszköz által igényelt korai (vagy az üzembe helyezés ideje alatt kiadott) beállítóparancsok küldésére is. Ha a visszatérési érték 0, az USB soros mag folytatja az üzembe helyezés műveletsorozatát. Bármely más érték a hívás megszakadását okozza, ekkor az USB-mag értesítést kap arról, hogy az adott eszközt egyik illesztőprogram sem kezeli.

A calc_num_ports függvény meghívásával határozható meg, hogy az adott eszköz hány kapuval rendelkezik. Csak olyan eszközöknél használható, amelyek képesek dinamikusan meghatározni a saját kapuikat. A visszatérési érték – bármi legyen is – felülbírálja az usb_serial_device_type adatszerkezet num_ports mezőjének a tartalmát. Meghívására a probe és az attach függvények hívása között kerül sor.

Az attach függvény meghívása az usb_serial adatszerkezet teljes feltöltése után történik. A helyi eszközindító műveletek indítása és a saját memóriában lévő adatszerkezetek lefoglalása ezzel a függvénnyel történik meg. A shutdown függvényt akkor szükséges meghívni, ha az eszközt eltávolították a rendszerből. Az eszközhöz helyileg lefoglalt memóriát ekkor kell felszabadítani.

A tty-réteg függvénymutatói a következők: open, close, write, write_room, ioctl, set_termios, break_ctl, chars_in_buffer, throttle és unthrottle. Ha visszagondolsz a tty-rétegről szóló cikkre („A tty-réteg” – a Linuxvilág 2002. szeptemberi számában), annak a rétegnek a kapcsán is pontosan ilyen nevű hívásokat tárgyaltunk – néhány eltérésről azonban nem szabad megfeledkeznünk. Először is mindegyikük átad egy mutatót az éppen használt usb_serial_port adatszerkezetre, valamint egyes függvények meghívására csak akkor kerül sor, ha valamilyen eseményt kell elindítani.

Az open függvény hívására egyedül az adott kapura vonatkozó első open() híváskor kerül sor, a további open() kéréseknél már nem. Az illesztőprogram által az adatok fogadásának megkezdéséhez igényelt urb-átviteleket ekkor kell megejteni, illetve az eszközre egyedileg jellemző üzeneteket is ilyenkor kell elküldeni. Ha bármilyen hiba történik, a hívás ennek azonosítójával tér vissza, egyébként 0-t adva jelzi a sikert.

A close függvényt az utolsó, a felhasználói területről induló close() híváskor kell meghívni. A folyamatban lévő urb-k ekkor véget érnek, és az eszköz által igényelt egyedi parancsok küldése is ekkor zajlik.

A write függvény pontosan úgy működik, ahogy a tty-rétegé is. A függvényhíváskor átadott adatokat a rendszer a megadott kapura továbbítja. Visszatérési értéke az eszköznek elküldött bájtok száma. Ne feledd, az eszköznek nem muszáj az összes, a felhasználó által elküldeni kívánt adatot elküldenie. Rövid írásra is sor kerülhet, feltéve, hogy az illesztőprogram értesíti erről a felhasználói oldalt. Az illesztőprogram szerkezete ezzel is egyszerűbbé válhat. Ha hiba történik, a hívás negatív értékkel tér vissza.

A write_room és a chars_in_buffer függvények szoros kapcsolatban állnak egymással. A write_room függvényt a tty-réteg hívja meg, segítségével azt kérdezi le, hogy az illesztőprogram éppen hány elküldendő bájtot tud fogadni. A chars_in_buffer függvény azt adja meg, hogy hány bájt várakozik még az eszköz felé történő elküldésre.

Az ioctl függvény hívására számos tty ioctl értékkel kerül/kerülhet sor. Ha az illesztőprogram az adott értéket nem tudja kezelni, akkor a -ENOIOCTLCMD értéket kell visszaadnia. Így lehetővé válik, hogy a tty-réteg megpróbálkozzon az alapértelmezett függvény használatával. A felhasználói területről leggyakrabban használt értékek közül néhányról a már említett tty-illesztőprogramról szóló cikkben bővebben is esik szó.

A set_termios függvénnyel adhatók meg a meghatározott kapu terminálbeállításai, többek között a sebesség, a folyamvezérlés és az adatbitek száma. A break_ctl függvény az adott kapu BREAK értékét adja meg. A -1 érték a BREAK Állapot be-, a 0 pedig a kikapcsolt állapotát jelzi. A throttle és unthrottle függvény a soros kapuról érkező adatok fogadásának felfüggesztésére és újraindítására használható.

urb visszahívó függvénymutatók

A read_int_callback, read_bulk_callback és write_bulk_callback függvénymutatókat az USB soros mag használja az ilyen típusú USB-végpontokhoz tartozó kezdeti visszahívók megadására. Ha az illesztőprogram nem ad meg tömeges adatmozgatásra szolgáló read vagy write visszahívót, a rendszer az általános célú visszahívókat használja. Általános read megszakítás-visszahívó függvény nincs, ha tehát az eszköz megszakításvégponttal rendelkezik, akkor ezt a visszahívót neked kell megadnod.

Az általános, tömeges adatmozgatást végző read visszahívó az urb révén kapott adatokat a kapu tty átmeneti tárába írja, ahonnan a felhasználói oldal read() hívással olvashatja ki őket. Az urb ezután visszakerül az eszközhöz. Ha az eszköz oldalán nincs szükség a kapott adatok bármilyen módon való értelmezésére, akkor azt javaslom, hogy ne írj új függvényt, hanem ezt használd. Az általános, tömeges adatmozgató write visszahívó jóval kisebb, és csak a tty-réteget „ébreszti fel” (ha az éppen „aludt”, vagyis az eszköznek továbbítandó adatokra várt).

Összegzés

Írásomban az USB-soros illesztőprogramok bejegyzésének és törlésének módját ismertettem, illetve a minden USB-soros illesztőprogram által kezelendő fő usb_serial_driver_type adatszerkezet legfontosabb jellemzőit mutattam be. A következő alkalommal részletesebben is foglalkozunk az USB-soros illesztőprogram réteg működésével, és egy példaprogramot is áttekintünk.

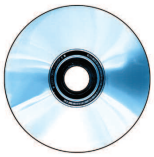
Köszönetnyilvánítás

Szeretnék köszönetet mondani mindazon programozóknak, akik az elmúlt évek során közreműködtek az USB-soros réteg fejlesztésében. Külön szeretném kiemelni ***Peter Berger*** és ***Al Borchers*** 2000 júliusában készült betölthető modul kódját, ami még ma is remekül szolgál.

Kapcsolódó anyag található a 46. CD Magazin/USB könyvtárában.

Linux Journal 2003. február, 106. szám

Greg Kroah-Hartman (greg@kroah.com)
Jelenleg a Linux USB és a PCI Hot Plug rendszermag felelőse. Az IBM-nél dolgozik, ahol számos, a Linux rendszermagjával kapcsolatos kérdéssel foglalkozik.



RAID és LVM alapfokon

Egy megoldás, amivel biztonságosan és rugalmasan kezelhetjük a merevlemezünket, hogy adatainkat minél nagyobb biztonságban tudjuk.

Mit is jelentenek ezek a rövidítések? A RAID nyers fordításban a merevlemezek hibatűrő tömbje. Számos fajtája létezik, nem is részletezem mindet. Ebben a cikkben elsősorban a programból megvalósított RAID-1 (software RAID-1, azaz tükrözés) változatról lesz szó. Ehhez a megvalósításhoz semmilyen különleges eszközre nincs szükségünk – használhattunk hozzá teljesen különböző merevlemezeket, illetve csak lemezrészeket is.

- RAID-0 (striping, csíkozás): minden adat egyszer kerül mentésre. Ha a tömböt alkotó merevlemezek bármelyike megsérül, az összes adat elvész, hacsak nem bízzuk méregdrága szakértőkre az adatok visszafejtését. Egyetlen tárterületként lehet majd a megadott eszközöket látni, és párhuzamosan ír, illetve olvas róluk, így a teljesítmény meghaladja egyetlen merevlemezét.
- A RAID-1 (mirrorring, tükrözés): minden adat mindkét vagy több merevlemezen (lemezrészen) is megtalálható lesz. Ha egy merevlemez (lemezrész) ép marad, nem vesznek el az adatok.
- A Logical Volume Manager (LVM): a lemezrészek magasabb szintű kezelését teszi lehetővé. Pontosabban ebben az esetben nem is lemezrészekről van szó, hanem úgynevezett kötetekről. Több fizikai lemezrész összekapcsolásával kötetekre oszthatjuk fel, amelyeket sokkal rugalmasabban lehet kezelni. Például ReiserFS-sel kombinálva a lemezrész leválasztása (`umount`) nélkül át lehet méretezni.

Gyakorlati megvalósítását a későbbiekben egy példán fogom szemléltetni, vagyis azt, hogy miként lehet elvégezni ezeknek az eszközöknek a beállítását. A programból megvalósított RAID-1-et és az LVM-et két 60 GB méretű merevlemezre raktam, mindezt az `initrd` használata nélkül, LILO indításkezelővel. A LILO elvileg csak a RAID-1-ről történő rendszerindítást támogatja, és azt is csak úgy, ha ugyanattól a szektortól kezdődnek az adott lemezrészek. Más RAID-módot is lehet azonban használni, de erről később még lesz szó.

Mire lesz szükségünk?

Legalább két merevlemezre lesz szükségünk, de természetesen több is lehet. Nem kell sem a típusuknak, sem a méretüknek megegyeznie, de ajánlott. A két merevlemez sem mindenképpen kötelező, csak kevesebbnek nincs értelme, mert ha az az egy meghibásodik, minden adat elvész. Minél újabb, lehetőleg 2.4.x változatú rendszermagunk legyen, benne RAID- és LVM- (és ReiserFS-) támogatással. Továbbá elkélnek a RAID (`mdadm`) és LVM (és ReiserFS-es) eszközöket kezelő programok, amiket a ***Kapcsolódó címekről*** tölthetünk le.

Beállítások

Mindkét merevlemezt három lemezrészre daraboltam fel: 1 GB-ot lapozóterületnek (swap), 1 GB-ot gyökérkönyvtárnak (autoraid), a többit pedig minden másnak, szintén `autoraid` azonosítóval (ID). Az azonosítót `cfdisk` vagy `fdisk` programmal is beállíthatjuk (hexában fd).

A második lemezrészre (`hda2`) Debian alaprendszert telepítettem, 2.4.20-as rendszermagot fordítottam rá, ami támogatja LVM- és RAID-eszközöket, a ReiserFS-t, az `ext3`-as fájlrendszert, továbbá mindent, amire még szükség lehet. Beállítottam a LILO-t, és újraindítottam az új rendszermaggal. Ha ez valakinél gondot okozna, a ***Kapcsolódó címek*** között megtalálható a kernel-HOWTO, amiben minden szükséges tájékoztatást megtalálhatunk hozzá. Az egyik RAID-eszközre, amin a gyökérkönyvtár (/) lesz, `ext3` fájlrendszert raktam, a másikra pedig LVM-et, arra pedig ReiserFS-t. Azért esett a ReiserFS-re a választásom, mert naplóz, és mindezt elég gyorsan is teszi, átméretezéshez pedig nem kell leválasztani (`unmount`), és külön rendszermagfolt (patch) sem kell hozzá, mint az `xfs` esetén. Természetesen bármilyen más fájlrendszer is használható. Következő lépésként feltelepítettem a szükséges programokat: `mdadm` (a RAID kezelésére), `lvm1.0.6`, `reiserfsprogs` és `e2fsprogs`.
Az LVM-et forrásból telepítettem, mert olyan helyen kell elhelyezni, ami nem kerül logikai lemezrészre (LV), például a */root/usr*-ben. Mindezt a következő parancsokkal értem el:

```
mkdir /root/usr;
./configure --prefix=/root/usr;
make;
make install
```

Az én esetemben a következők kerültek a logikai lemezrészekre: a */home* */usr* */tmp* és a */var*. Hogy futtatni is tudjam a programokat, a ***.profile*** fájlba (vagy a ***.bash_profile***-ba), a `PATH` változóba be kellett rakni a */root/usr/sbin* elérési utat. Ahhoz, hogy a programok a függvénykönyvtárakat (library) is megtalálják, a */etc/ld.so.conf* fájlba a következő sort kellett beírnom (Debian esetén): */root/usr/lib*, majd a `ldconfig` parancsot futtattam. Ezenkívül egy indítófájl is szükséges, amit kézzel is létrehozhatunk (megtalálható a CD-mellékleten), vagy kiszedhetjük Linux-terjesztésünk hivatalos változatából (természetesen a megváltoztatott elérési útnak (*/root/usr*) megfelelően át kell írni). Ennek a fájlnak alapvetően az a feladata, hogy elindítsa az LVM-et. Két fájlt kell elindítania:

```
vgscan;
vgchange -a y
```

Ez a mi esetünkben:

```
/root/usr/sbin/vgscan;
/root/usr/sbin/vgchange -a y
```

Ahhoz, hogy ez a parancsfájl a rendszerindítás során, Debian GNU/Linux esetén el is induljon, ***S25lvm*** néven egy közvetett hivatkozást kell elhelyeznünk a */etc/rcS.d* könyvtárba (ezt az `lvm-common` csomag telepítése esetén meg is teszi). Arra vigyázni kell, hogy általában `mdadm-raid` közvetett hivatkozása

2. lista Az fstab

```
# /etc/fstab: static file system information.
#
# <file system>    <mount point>  <type>     <options>                  <dump>    <pass>
/dev/md0           /              ext3       defaults,errors=remount-ro 0         1
#/dev/hda2         none           swap       sw                         0         0
/dev/hdc2          none           swap       sw                         0         0
proc               /proc          proc       defaults                   0         0
/dev/lvgroup/usr   /usr           reiserfs   defaults                   0         2
/dev/lvgroup/tmp   /tmp           reiserfs   defaults,noexec            0         2
/dev/lvgroup/var   /var           reiserfs   defaults                   0         2
/dev/lvgroup/home  /home          reiserfs   defaults                   0         2
```

3. lista Példa a resize-ra

```
#!/bin/sh
#használat: resize device új_meret_megabájtban
set -e
if [ $# = 2 ] ; then
        /sbin/lvextend -L${2}M "${1}"
        # a 256 mágikus szorz
        # az 1MB/4KB-os ReiserFS blokkméret
        FOURKBLOCKS='/usr/bin/expr ${2}
        ➥'*' 256 '
        /bin/mount -n "${1}"
        ➥-oremount,resize=${FOURKBLOCKS}
        /bin/echo '++++ SIKER ++++'
        /bin/df -k "${1}"
else
        /bin/echo "Használat: ${0}
        ➥device_neve újméretmegájtban"
fi
```

is ***S25***-tel kezdődik, így az LVM indul el hamarabb. Ha olyan RAID-eszközt is használni szeretnénk, ami nem rendszerindítás közben éled fel (vagyis típusa nem `linux raid autodetect`), akkor az LVM-et később indítsuk, például ***S26lvm*** legyen a közvetett hivatkozás neve. Ha mindezzel elkészültünk, akár használhatjuk is a LVM-et, de előtte a RAID-eszközt létre kell hozni. Az LVM állapotáról a */proc/lvm* könyvtárban lévő fájlokból nyerhetünk értékes adatokat.
Az `mdadm` csomagból történő feltelepítése Debian esetén elég egyszerű:

```
apt-get install mdadm
```

Először is létrehoztam egy RAID-eszközt:

```
mdadm --create /dev/md0 --level=1 -n 2 /dev/hda1
➥/dev/hdc1
```

Ezután a */dev/md0* elérhető. A kapcsolók jelentése a következő:

- `--level=1`: RAID-1 módban hozza létre;
- `-n 2`: 2 disk lesz a RAID-eszközben;
- */dev/hda1* */dev/hdc1*: milyen eszközök alkotják az eszközt.

Akik csupán a RAID-eszközök képességeit kívánják kipróbálni, egyszerűen a hurokeszközön (loop) keresztül is befűzhetnek fájlokat, majd az így kapott eszközöket használhatják egy RAID-tömbhöz.

A merevlemezek összehangolása azonnal elkezdődik, ezt a `cat /proc/mdstat` parancs révén követhetjük figyelemmel. Nem szükséges megvárni az összehangolás végét, közben is használhatóak a merevlemezen lévő adatok. A másik eszközt is létrehoztam az előzőhöz hasonlóan:

```
mdadm --create /dev/md1 --level=1
➥-n 2 /dev/hda3 /dev/hdc3
```

Az összehangolás gyorsaságát a */proc/sys/raid/speed_limit_(min|max)* fájlokban kilobájtokban kell megadni.
Most következik a munka az LVM-mel. Logikai lemezrészt csak a */dev/md1*-re raktam, a gyökérkönyvtárként használt másik eszköznek állandó lesz a mérete. A létrehozandó köteteket csoportokba (Volume Group) kell rendezni. Egy merevlemezen több ilyen csoportot létrehozhatunk, illetve egy csoport is lehet több merevlemezen. Legelőször a `vgscan` programot futtattam, hogy elkészítse a */etc/lvmtab* és a */etc/lvmtab.d* fájlokat:

```
# vgscan
```

Ebben a szakaszban még egyetlen ilyen csoport sincs a gépen, de ezeknek a fájloknak létezniük kell, hogy folytatni lehessen a munkát. A következő két parancs egy fizikai lemezrészt (Physical Volume) és egy ***Logical Volume Group***-ot hoz létre:

```
# pvcreate /dev/md1
# vgcreate lvgroup /dev/md1
```

Az utóbbi neve tetszőleges lehet, ebben az esetben ez `lvgroup`. Itt több különböző beállításokat adhatunk meg, például a logikai, illetve fizika lemezrészek legnagyobb számát, és az úgynevezett `extentsize`-t. Alapértelmezés szerint ez 32 MB, ami megadja, hogy egy kötet nagysága csak a 32 MB többszöröse lehet. Az így kezelhető legnagyobb méret 2 TB. Erről a leírásban kimerítően olvashatunk. Én az alapbeállításokat használtam.
Az előállított csoportot tovább bontottam logikai lemezrészekre:

```
# lvcreate -L 2000M -n usr lvgroup
# lvcreate -L 4000M -n var lvgroup
# lvcreate -L 1200M -n tmp lvgroup
# lvcreate -L 10000M -n home lvgroup
```

Több ilyen lemezrészt is létre lehet hozni, amennyire csak szükség van. A `-L` kapcsoló után kell megadni a méretet, `-n`

után egy tetszőleges nevet, amin később majd hivatkozni lehet rá. Hogy a frissen elkészített logikai lemezrészeket használatba tudjuk venni, el kell indítani a kezelőprogramot:

```
# vgchange -a y
```

Mostantól a */dev/lvgroup/* (*usr, var, ...*) fájlok ugyanúgy érhetők el, mint bármely merevlemezeszköz, tehát rakhatunk rá fájlrendszert is. A gyökérlemezrészre (/) `ext3` fájlrendszert, a többire pedig ReiserFS-t telepítettem.

```
# mkfs.ext3 /dev/md0
# mkfs.reiserfs /dev/lvgroup/usr
# mkfs.reiserfs /dev/lvgroup/var
```

Ehhez hasonlóan a többi lemezrészre is létre kell hozni a fájlrendszert. A */mnt* könyvtárba raktam össze a leendő rendszert:

```
# mount /dev/md0 /mnt
# cd /mnt
# mkdir usr home var
# mount /dev/lvgroup/usr /mnt/usr
# mount /dev/lvgroup/var /mnt/var
# mount /dev/lvgroup/home /mnt/home
```

A */tmp* könyvtárral nem fontos foglalkozni, úgyis üres. Következő lépésként a teljes rendszert átmásoltam. Ezt a `cp`-vel vagy a Mindight Commanderből is meg lehet tenni, de ehhez hasonlóan is:

```
# tar -cf - --exclude /proc --exclude /mnt /
➥| (cd /mnt && tar xf - )
```

Gondolom, ezen nincs mit magyarázni: a gyökérkönyvtárból (/) mindent átmásoltam a */mnt*-be, a */proc* és */mnt* kivételével.
Át kell állítani a rendszert, ami leginkább a *lilo.conf*-ot és az `fstab`-ot érinti, miként az az 1. és 2. listán (a listák megtalálhatóak a 46. CD Magazin/RAID könyvtárában) is látható.
Mivel a `hda2` egy működő rendszeren csücsül, nem érdemes még lapozóterületként használni, csak miután a RAID-eszközökről jól működik a rendszer. A `hdc2` úgyis üres, ezért rögtön használhatjuk:

```
mkswap /dev/hdc2 && swapon /dev/hdc2
```

A rendszerindítás során a vezérlés az adott merevlemez rendszerindító területére (MBR) adódik át (és nem közvetlenül a RAID-eszközre), ezért ide kell telepíteni egy indításkezelőt (boot manager), ami a vezérlést egyszerűen átadja a következő lemezrészre, ami képes egy operációs rendszer indítására.
Mindezt mind a kétszer (vagy annyiszor, ahány merevlemezből áll a RAID-eszköz) meg kell tenni:

```
install-mbr /dev/hda;
install-mbr /dev/hdc;
```

Ezek után futtattam a LILO programot (`lilo -C /mnt/etc/lilo.conf`), és újraindítottam a gépet. A CD-mellékleten megtalálható lesz a `dmesg` parancs kimenete, vagyis az, hogy miként is kellene a rendszerindítás során kinéznie.
A *lilo.conf*-ból, miután a rendszer sikeresen elindult, a */mnt* részt érdemes a megadott fájlok elérési útjából kitörölni, és újra lefuttatni a LILO-t. Ezzel készen is vagyunk. Szükségünk lehet még a futás közbeni átméretezésre, amit 3. *listán* található kis programmal lehet megtenni.

Mi történik, ha nem a RAID-1-módot szeretnénk használni?

Próbaképpen egy 1,2 GB-os merevlemezre Debian GNU/Linux-ot raktam fel oly módon, hogy két 500 MB méretű lemezrész alkotott egy RAID-1-eszközt. 200 MB-ra került fel az alaprendszer, majd ebből lett a lapozóterület. Egyetlen merevlemezre rakni a RAID-1-eszközt alkotó lemezrészek mindegyikét, alapjában véve értelmetlen, de ebben az esetben azért tettem kivételt, hogy kipróbáljam, más szektorról indulva is képes-e elindulni a rendszer a LILO segítségével. A beállítás a fenti példától annyiban különbözik, hogy az alaprendszer telepítése után a RAID-eszközt a `hda1-hda2` lemezrészekre raktam. A rendszer így is elindult, de ebből gondok is származhatnak. Ugyanis a RAID-1-et alkotó lemezrészek más szektoron kezdődnek, akkor az egyik merevlemez meghibásodása eseten a rendszer valószínűleg nem fog elindulni. De kombinálhatjuk is az egyes RAID-fajtákat. Kevésbé fontos adatok esetén vagy ahol a gyorsaság számít, használhatunk RAID-0-t. A RAID-0-megoldás esetén mindkét merevlemezen egyszerre dolgozik, az adatokat elosztva tárolja a két lemezen, így növeli az írás, illetve olvasás sebességét. Például ezen lehet a rendszer vagy az olyan adatok, amiknek az elvesztése nem nagy szerencsétlenség, inkább a sebesség számít. A rendszer és a fontosabb adatok lehetnek RAID-1-eszközön; egy olyan lemezterület pedig, amelyen nagyméretű videofájlokat tárolunk, a RAID-0 eszközön.
A RAID-5, illetve RAID-10 módot is lehet arra a célra használni, hogy róla induljon a rendszer, csak ebben az esetben nem megfelelő a biztonság, mivel az egyik merevlemez meghibásodása esetén a rendszer valószínűleg nem fog megfelelően elindulni. Ilyenkor mindenképpen `initrd`-t kell használni, aminek az ismertetése túlmutat e cikk keretein, viszont a ***Kapcsolódó címek*** között elegendő tájékoztatást lehet találni ezzel kapcsolatban is.
Végezetül pár szót a helyrehozásról. Ha a rendszer rendszermagfrissítés után vagy más hiba miatt nem indul, egyszerűen helyrehozható, ha kéznél van egy `knoppix` CD. Röviden ez egy Debianra épülő egylemezes Linux. Erről a korongról indítva a rendszert, az rögtön felismeri a RAID- és LVM-eszközöket, és akár grafikus felületen is rendbe hozhatjuk, mivel minden szükséges eszköz megtalálható rajta.
Köszönet a ZiB tagjainak, és elsősorban ***Vitéz Gábor***-nak a sok segítségért.

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

KAPCSOLÓDÓ CÍMEK

Knoppix ➲ http://www.knoppix.org/
RAID ➲ http://linux.oreillynet.com/pub/a/linux/2002/12/05/RAID.html
➲ http://mirror.dox.hu/LDP/HOWTO/Root-RAID-HOWTO.html
➲ http://mirror.dox.hu/LDP/HOWTO/Software-RAID-HOWTO.html
LVM ➲ http://www.sistina.com/products_lvm_download.htm
➲ http://mirror.dox.hu/LDP/HOWTO/LVM-HOWTO/index.html
ReiserFS ➲ http://www.namesys.com/download.html
Kernel ➲ http://www.tldp.org/HOWTO/Kernel-HOWTO.html

Anyagtulajdonságok meghatározása (3. rész)

Sorozatunk előző részeit figyelemmel kísérve mostanra már különféle tárgyakat tudunk megalkotni a Blender segítségével, így hát elérkezett az ideje, hogy tárgyaink anyagát is megtanuljuk létrehozni.

A Blenderben, miután a tárgyak végleges formáját kialakítottuk, egy piros gömböt ábrázoló ikon jelzi az anyagtulajdonságokat meghatározó felhasználói felületet, amit az F5 billentyűvel érhetünk el. Jelöljünk ki egy objektumot, és az egérmutatót vigyük az alsó felületre, itt nyomjuk meg az F5 billentyűt – ennek hatására előbukkan egy üres felület. Nem kell megijedni, hiszen a tárgynak egyelőre nincsen anyaga, így a tulajdonságai sem jelenhetnek meg. A felület felső szélén azonban két nyilacskát találunk: az egyik lefelé mutat, míg a másik felfelé. A tőlük balra lévő vízszintes fehér vonalra kattintva válasszuk ki a menüből az ***ADD-NEW*** menüpontot. Jelenleg nem is tudnánk mást választani, de amikor már lesznek elkészített anyagaink, akkor innen választva is hozzárendelhetjük majd a tárgyakhoz az anyagtulajdonságokat. Szerencsére, ha egyszer sikerül tökéletesre megalkotni egy tárgyat, anyagot vagy bármilyen Blenderben elkészíthető dolgot, a későbbiekben már nem kell megismételnünk a folyamatot, hiszen a program lehetőséget nyújt arra, hogy bármilyen létrehozott ***.blend*** kiterjesztésű (ez a Blender által tárolt jelenetek kiterjesztése) állományból a pillanatnyi jelenetbe összetevőket töltsünk be. Ezt a SHIFT-F1 billentyűkkel valósíthatjuk meg, ennek hatására megjelenik egy állománykiválasztó ablak. Itt egy ***.blend*** állományra állva, annak tartalmát láthatjuk, és ebből a ***Load Library*** gomb használatával tetszőleges elemet adhatunk hozzá a pillanatnyi jelenethez.

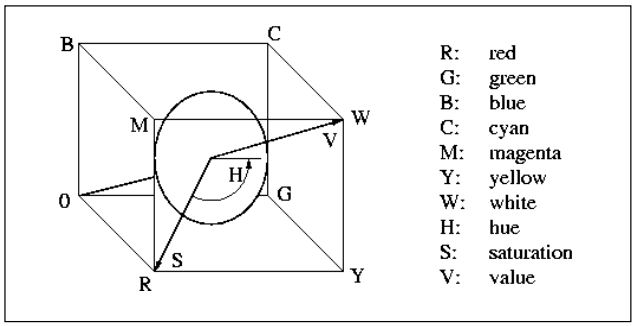
Az *1. képen* a felület bal oldalán látható a pillanatnyi anyag kinézete, vagyis az, amit majd a kép kiszámolása során is viszontláthatunk. Efölött, a képen 1-es számmal jelölt gombok segítségével beállíthatjuk, hogy az anyag meghatározása közben milyen megjelenítéssel szeretnénk látni beállításaink eredményét. A számított képtől jobbra három gombot találhatunk egymás alatt ***RGB***, ***HSV*** és ***DYN*** felirattal, amik azt határozzák meg, hogy a színeket milyen színtérben szeretnénk látni. Az ***RGB*** a már megszokott vörös, zöld, kék összetevőkre bontott értékeket jelzi, a ***HSV*** a Hue, Saturation, Value megfelelője, míg a ***DYN*** a Blender játékmotorjának tárgyalásakor kap majd jelentőséget, az anyag fizikai tulajdonságainak meghatározásakor. Az előbbiek közül több szót érdemel a ***HSV*** színtér leírása. A `H` érték határozza meg a színt, amit fokban fejezzük ki. A 0 foknál található a piros, 120 foknál a zöld és 240 foknál a kék szín. A ***HSV*** színtérben az `S` érték a szín tisztaságát határozza meg, minél kisebb ez az érték, annál tisztább színnel van dolgunk. Az utolsó érték a `V`, ez határozza meg, hogy az adott szín mennyire lesz világos. A pasztellárnyalatokra a nagyobb `V` érték jellemző. A *2. képen* egymásra vetítve láthatjuk a két színtér közötti összefüggéseket, itt azonban a szokásostól eltérően a kék színt találhatjuk a nulla foknál.

Az 1. képen 2-es számmal jelölt három színnel határozhatjuk meg az anyag színeit. A `Mir` a visszavert fény színe lesz, a `Spec` a felületi csillanások színét adja, míg a `Color` az anyag alapszínét határozza meg. Mindhárom összetevő változtatására alkalmas a színeket jelző téglalapok melletti három gördítősáv, ezek értékei természetesen az ***RGB*** vagy ***HSV*** színtérben értelmezendők, a korábbi beállításainktól függően. A színeket jelző terület alatt kaptak helyet a további anyagtulajdonságok. A `Spec` értékkel befolyásoljuk az anyag fényvisszaverő képességét, a `Hard` az anyagkeménységet befolyásolja, míg a `SpTr` értékkel egy átlátszó anyagnál azt határozhatjuk meg, hogy a csillogó területek mennyire váljanak átlátszatlanná. A további tulajdonságokat a `Ref`, az `Alpha`, az `Emit` és az `Amb` értékekkel állíthatjuk be. Az előbbi sorrendnek megfelelően ezekkel határozhatjuk meg az anyag tükrözőképességét (itt az egész tárgyra vonatkozó értéket állítjuk be), az átlátszóságát, a saját fénykibocsátó képességét és azt, hogy a környezeti megvilágítás mekkora hatással legyen a tárgyra.

A 3-as számmal jelölt területen lévő gombok közül a ***Traceable*** azt eredményezi, hogy a tárgy megjelenjen a rávetődő fényben, míg a ***Shadow*** engedélyezi az árnyékok számítását is. A ***Shadeless*** kapcsolóval kikapcsolhatjuk az anyagra történő árnyékolást, míg a ***Wire*** kapcsoló segítségével a Blender csak a tárgyat alkotó háromszögek éleit számolja ki. A ***VCol Light*** segítségével a tárgy pontjaihoz rendelt színértékek helyi fényforrásként vesznek részt a számolásban, kissé módosítva a tárgy anyagának árnyékolását. A ***VCol Paint*** kapcsoló engedélyezi, hogy az alapértelmezett (a ***Color*** tulajdonság által meghatározott) szín helyett az egyes tárgypontokhoz rendelt színek érvényesüljenek. A ***ZTransp*** segítségével érhetjük el, hogy az anyag valóban átlátszó legyen és a tárgy mögött lévő objektumok is megjelenjenek. Az ***Env*** kapcsoló hatására az anyag semmilyen tulajdonsága nem jelenik meg számolás közben, vagyis a környezet részeként szerepel a jelenetben. Ezzel a beállítással tudjuk majd a háttérképen szereplő kémény mögött elrepülő űrhajót létrehozni, anélkül, hogy a kémény anyagát pontosan meg kellene határoznunk. Az ***OnlyShadow*** kapcsoló hatására a tárgy átlátszóságát a rávetődő árnyékok fogják meghatározni, így készíthetünk olyan tárgyakat is, amik csak árnyékként szerepelnek a jelenetben.

Talán kedves olvasóim is észrevették, hogy a fenti ismertetőből kimaradt egy fontos gomb leírása, amin a ***Halo*** felirat látható. Következzen tehát ez a jelentős lehetőség, amivel különféle csillanásokat hozhatunk létre a tárgyon, fényhatásokat határozhatunk meg, és természetesen ezzel az eszközzel készíthetünk majd látványos, a kamera lencséjén megcsillanó fényforrásokat is. Tehát először is be kell kapcsolni ezt a szolgáltatást, majd észrevehetjük, hogy a korábban tárgyalt értékek helyén teljesen más elnevezések jelennek meg. Először vegyük szemügyre a ***Halo*** gomb fölött megjelenő újabb lehetőségeket. A ***Star*** hatására a csillanásnak

1. kép Anyagtulajdonságok a Blenderben



2. kép Az RGB és HSV színtér

SHIFT-F1	Betöltés másik állományból
F5	Anyagbeállítások
F12	Kép számítása
F11	Számított kép megjelenítése/elrejtése

csillaghoz hasonló formát adhatunk, míg a ***Rings*** alkalmazásával a csillanás körül az autók reflektora körül látható gyűrűkhöz hasonló fénygyűrűket állíthatunk elő. A ***Lines*** szintén jelentősen befolyásolhatja a végeredményt, hiszen így a csillanásból még külön fénysugarak indulnak, amit leginkább ahhoz a látványhoz hasonlíthatnék, amit egy szemüvegen keresztül nézve láthatunk az utcai lámpák fényében. Miután az elképzeléseinknek megfelelően kiválasztottuk a megfelelő hatásokat, térjünk vissza egy kis időre az 1. képhez. Ezen a képen a 2-es számmal jelölt területen találjuk meg a csillanás színeit meghatározó beállításokat. A `Mir`, a `Spec` és a `Color` feliratok helyett most a csillanás előbb tárgyalt értékeit állíthatjuk be a ***HSV*** vagy a szokásos ***RGB*** színtérben dolgozva. A színek meghatározása után ezek alatt beállíthatjuk a hatás átlátszóságát, a csillanás erősségét és azt, hogy milyen mértékben növelje a Blender a fényhatás erősségét. A színek alatt adhatjuk meg a gyűrűk és a sugárirányú fények számát, továbbá azt, hogy a csillanásban megjelenő csillagnak mennyi ága legyen. Ezeknek felelnek meg a ***Rings***, a ***Lines*** és a ***Star*** címkék mellett szereplő számok. A természetesebb hatás eléréséhez nagy segítség, hogy a `Seed` értékkel a számítások során véletlenszerűséget adhatunk a megjelenítéshez. Ennyi beállítási lehetőség ismeretében már képesek leszünk egy-egy látványosabb hatást létrehozni, például olyat, ami az éjszakai fényekhez hasonlítható. Fentebb azonban megemlítettem, hogy a kamera lencséjén megcsillanó fényeket is ezzel az eszközzel modellezhetjük. Itt az ideje tehát, hogy az eddig használt felület közepének felső részén található ***Flare*** kapcsolót is kipróbáljuk. Ennek bekapcsolásakor rögtön megváltozik a hatás jellege, ugyanis most már nem csupán egy pontszerű fényforrásból származik a csillanás, hanem pontosan azt láthatjuk a felület bal oldali megjelenítő ablakában, amit egy fény felé forduló fényképezőgép lencséjén figyelhetünk meg. Ha esetleg még mindig elégedetlenek lennénk az eredménnyel, nézzük meg, milyen további beállítási lehetőségeink adódnak. A színek beállítása alatt találhatjuk meg az előzőekben még nem taglalt gombokat. Ezek közül az első a ***Flare Size***, amivel azt állíthatjuk be, hogy a forrástól távolodva hányszorosára növekedjen (1-nél nagyobb érték esetén) vagy csökkenjen (1-nél kisebb értéket beállítva) a csillanások mérete. Mellette található a ***Sub Size*** érték, amivel a távolabb látható részek méretét határozhatjuk meg. A ***Flare Boost*** változó segítségével a forrástól távolodó részek erősségét állíthatjuk be, míg a legalsó gombok közül a ***Flare Seed*** a természetesebb véletlenszerűség létrehozására alkalmas, a ***Flares*** értékkel pedig a kiszámítandó távoli gyűrűk számát határozzuk meg. Jegyezzük meg, hogy amelyik tárgyhoz majd hozzárendeljük ezt az anyagot, annak minden pontjában egy ilyen helyi fénycsillanás jelenik meg. Ha azt szeretnénk, hogy a kamera lencséjén csillanjon meg a fény, hozzunk létre például egy síklapot a ***Főmenü-Add-Mesh-Plane*** menüpontok segítségével, és ennek két egymás melletti sarkát hozzuk a másik két ponttal fedésbe. Ezzel elértük, hogy egymás mellett ugyan két pont van, de a pontos fedés miatt a számolás során csak egynek látjuk majd. Miután a csillanást szolgáltató anyagot hozzárendeltük a síkhoz, méretezzük át a tárgyat, és az egyik végét helyezzük el a kamera látószögének a közepében. Ezután ismét kapcsoljunk vissza pontszerkesztő módba, és a síklapból kialakított vonal másik végét helyezzük el minél pontosabban a fényforrás közepében. Ezzel készen is van a látványos lencsecsillanás, amit tekinthetünk meg úgy, hogy a kamerát egy kissé elfordítjuk a fényforrástól, majd az F10 billentyű megnyomásakor előbukkanó felületen a ***Render*** gombot használjuk. A változás akkor lesz látható, ha visszafordítjuk a kamerát, és egy újabb képet számoltatunk ki a Blenderrel. Jegyezzük meg, hogy a számolást a továbbiakban az F12 billentyűvel is elindíthatjuk, illetve az F11 billentyűvel elrejthetjük a kiszámított képet mutató ablakot.
Miután meghatároztuk a tárgy anyagának tulajdonságait, a Blenderrel felületi mintázatokat is készíthetünk, amikkel tovább növelhetjük a tárgyak életszerűségét, terjedelmi okokból azonban ennek módját a következő hónapban ismertetem. Végezetül ismét táblázatban foglaltam össze az előbbiekben ismertetett billentyűket és hatásaikat.

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

Lapozás, szakaszolás, Intel Pentium

Sorozatunk előző részében a memóriagazdálkodással kezdtünk el foglalkozni, most tovább mélyítjük lapozással kapcsolatos ismereteinket, majd bemutatunk egy másik fontos virtuálismemória-kezelő „eljárást”, a szakaszolást (segmentation).

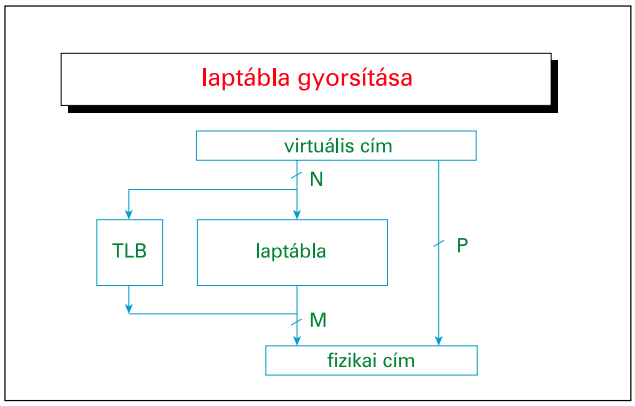
Sorozatunk előző részében valahol ott hagytuk abba, hogy a lapozás megvalósításának nagy részét maga az eszköz, pontosabban a processzorlapkán elhelyezett memóriakezelő egység (MMU) végzi. Az operációs rendszernek „csupán” annyi a feladata, hogy laphiba esetén a kért lapot megkeresi a lemezen, majd betölti egy általa kiválasztott lapkeretbe.
Részletesen szóltunk a laptáblaról is, vagyis arról a táblázatról, amelyből kiolvashatjuk, hogy egy lap benn van-e a fizikai memóriában, és ha igen, akkor melyik lapkeretben. Másképp fogalmazva: a laptábla egy leképezés a lapok és a lapkeretek között. (Emlékezzünk vissza, hogyha egy lap nem volt benn, tehát laphiba lépett fel, akkor az operációs rendszerhez megszakítási kérelem érkezett, és ennek hatására olvasta be a kívánt lapot a lemezről).

Többszintű laptáblák

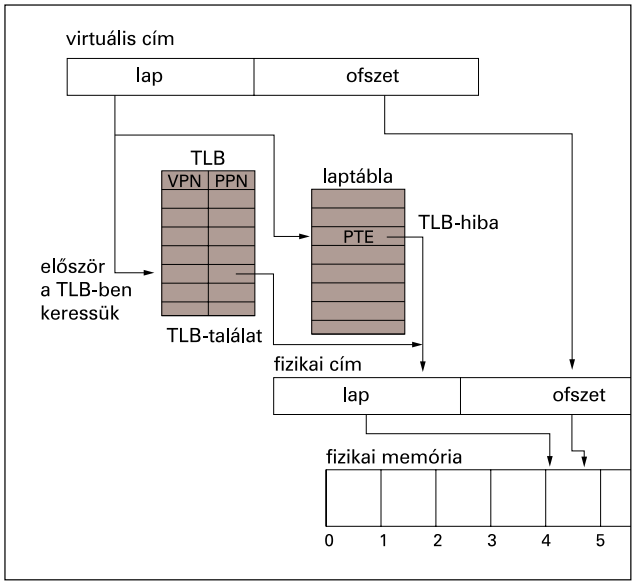
A lapozás akkor és csakis akkor lehet hatékony megoldás, ha a laptábla gyorsan elérhető, azaz a lapok és lapkeretek közötti leképezést a lehető leghamarabb el tudjuk végezni. Ugyanis minden egyes utasítás alkalmával legalább egyszer (de általában többször) a memóriához fordulunk, így elengedhetetlen, hogy a laptáblából legfeljebb pár nanoszekundum alatt ki tudjuk nyerni a szükséges adatot.
Jó megoldásnak tűnhet, ha az egész laptáblát bepakoljuk a fizikai memóriába. Egy 32 bites virtuális címtér esetében 4 KB-os lapmérettel a laptábla körülbelül 4 MB-ot foglal el, ami a mai viszonyok között nem vészes érték. Más a helyzet a 64 bites számítógépek világában. Ha az egész címteret ki szeretnénk használni, akkor még 4 KB-os lapméret esetében is a laptábla „csekély” egymillió gigabájtot foglalna el. Lehet, hogy a nem is olyan távoli jövőben ez az érték nem számít majd nagynak, de jelenleg még nincs lehetőségünk arra, hogy csak a laptáblának gigabájtok százezreit ajándékozzuk. Emiatt az ilyen rendszerekben úgynevezett invertált laptáblákat alkalmaznak. A módszer lényege dióhéjban annyi, hogy nem az összes laphoz, hanem csak a fizikai memória lapkeretei számára tartunk fenn egy bejegyzést, ami megmondja, hogy az adott helyre éppen melyik program melyik lapja van betöltve. Az invertált laptábla jóval kevesebb helyet foglal, viszont a virtuális cím fizikaira való átalakítása jóval bonyolultabb művelet, ezáltal lassabb is, ezért ennél feltétlenül szükség van az asszociatív memória használatára (lásd a következő bekezdésben).
Számos rendszer azonban nem tartja benn állandóan az egész laptáblát a memóriában, inkább úgynevezett többszintű laptáblát alkalmaz. Ez azt jelenti, hogy a laptábla azon részei, amelyekre épp nincs szükség, nem kerülnek be a központi tárba. Példaként vegyünk egy olyan alkalmazást, amelyik 10 MB memóriát használ. Ebből 3 MB a programkód, 4 MB az adat és az utolsó 3 MB pedig a vermet képezi. Ebben az esetben a verem alja és az adatok „vége” között egy jókora kihasználatlan terület marad. Az e részre eső lapok laptáblabejegyzéseit így felesleges a memóriában tartanunk.

Asszociatív memória

A lapozással az a legnagyobb gond, hogy teljesítménycsökkenéssel jár. Vegyünk egy egyszerűbb utasítást, például az egyik regiszter tartalmának átmásolását egy másik regiszterbe. Ha nem használnánk lapozást, akkor az egész művelet elvégzése közben csak egyszer kellene a memóriához fordulnunk (amikor kiolvassuk belőle a pillanatnyi utasítást). A lapozásnál viszont szükségünk van a laptáblára is, ennek elérése további memóriahivatkozással jár. Kellett valami, aminek a segítségével lefaraghatunk a lapozás használatából eredő sebességromlásból.
A megoldás meglelésében az az észrevétel segített, hogy a legtöbb hivatkozás mindig a lapok csak egy kis hányadára érkezik, a többire viszonylag ritkán van szükségük a programoknak futásuk során. Ez azt jelenti, hogy a laptábla egyes bejegyzéseire gyakran, a többire meg csak ritkán van szükségünk.
Ezt használva ki az MMU-t egy kis gyorstárral egészítették ki, ahová betölthetők a laptábla gyakran használt bejegyzései. Ezt nevezzük TLB-nek (Translation Lookaside Buffer, azaz címfordítási gyorstár), más néven asszociatív memóriának. A TLB-ben található bejegyzések szerkezete megegyezik a laptáblával, azzal a különbséggel, hogy ki lett egészítve egy bittel, ami megmondja, hogy az adott bejegyzés érvényes adatokat tartalmaz-e vagy sem.
Amikor kérés érkezik az MMU-hoz, először az asszociatív memóriában próbálja megkeresni az adott laphoz tartozó bejegyzést. Ha sikerrel jár, a laptáblához való fordulás nélkül is el tudjuk végezni a műveletet. Ha viszont a keresett bejegyzés nincs a TLB-ben, a laptáblában muszáj megkeresnünk. Ebben az esetben az MMU kidob egy bejegyzést a TLB-ből, és a helyére betölti azt, amelyikre éppen szükségünk volt. (A helyzet azonban nem ilyen egyszerű, mivel az MMU csak a memóriában lévő bejegyzéseket tudja az asszociatív memóriába felvenni, így ha az addig lemezen volt, akkor előbb gondoskodnunk kell a memóriába történő átmásolásáról is.)
Ne felejtsük el, hogy ez az egész az eszköz szintjén zajlik, magyarán az operációs rendszer nem szólhat bele a megvalósításukba. Az utóbbi években azonban ezt az elgondolást egyre több helyen kezdik megváltoztatni. Például az újabb Alphák és sok más RISC processzoros kiépítés esetén szinte az egész lapkezelés programból megvalósítottan működik.
A programalapú feldolgozás sebessége ugyan elmarad valamennyire az eszközszitű kezeléstől, mindezek ellenére sokat nyerhetünk ezzel a megoldással. Először is az MMU felépítése egyszerűbb lesz (mivel kevesebb feladatot kell ellátnia), így a processzoron több hely marad más hasznos (teljesítménynövekedést elősegítő) elem elhelyezésére, mint például gyorstárra.
A másik fontos dolog, hogy az operációs rendszer döntheti el, hogy a laptábla milyen bejegyzései kerüljenek az asszociatív



Ha nem használnánk lapozást, akkor a virtuális címek megegyeznének a fizikaiakkal, így semmiféle átalakítást sem kellene végeznünk (P), egyébként a laptábla végzi el az átalakítást (N). A laptáblát kiválthatja a TLB, abban az esetben, ha tartalmazza a keresett bejegyzést kellene.



A képen a TLB működését láthatjuk. A laptábla keresendő sorát először a TLB-ben nézzük meg, ha ott nem találjuk, akkor fordulunk csak a laptáblához.

memóriába. Ezáltal lehetőség nyílik a TLB-hibák csökkentésére (azaz minél kevesebbszer forduljon elő, hogy a keresett bejegyzés nincs a TLB-ben), például úgy, hogy megpróbáljuk kitalálni, melyik lap lesz a legvalószínűbb, amelyre az adott programnak szüksége lesz. Ha például egy folyamat éppen egy másik program osztott eljárását készül meghívni, akkor sejteni lehet, hogy hamarosan a másik folyamat által használt lapokra is szükség lesz.

A tapasztalat azt mutatja, hogyha az asszociatív memória elég nagy, azaz körülbelül 64 bejegyzést is képes tárolni, akkor a programból történő kezelés gyakran sokkal hatékonyabb lehet a hagyományosnál.

Lapcserélési algoritmusok

Akár programból megvalósítva, akár eszközszinten megy végbe a TLB kezelése, az operációs rendszernek kitüntetett szerepe van a lapozás megvalósításában, mégpedig a lapok a lapkeretekbe való betöltésében.

Egy lapot akkor kell valamelyik lapkeretbe betöltenünk, amikor laphiba lép fel, azaz a keresett lap nincs a memóriában. Ebben az esetben helyet kell csinálnunk az új lapnak, vagyis ki kell szedni egy lapot a memóriából. Azt az elvet, aminek alapján az operációs rendszer kiválasztja azt a lapot, aminek távoznia kell, lapcserélési algoritmusnak nevezzük.

Kulcsfontosságú, hogy jó lapcserélési algoritmussal legyünk megáldva, különben programjaink futása elviselhetetlenül lassú lesz. Na igen, de mit is jelent az, hogy „jó”? Az a jó algoritmus, amelynek használata során a legkevesebb laphiba lép fel, azaz minél kevesebbszer kell „bántani” a merevlemezünket. (Nagyon fontos dolog, hogy a memóriában lévő lapok tulajdonképpen egyfajta másolatok, az eredetiek ugyanúgy megtalálhatóak a lemezen, mint a többi. Tehát egy memóriában lévő lap eldobásán nem mást értünk, minthogy felülírjuk egy másik, eddig kizárólag csak a lemezen tárolt lappal. Természetesen, ha egy lap tartalma megváltozik, miközben a memóriában tartózkodik, eldobása előtt a lemezen lévő eredeti példány frissítéséről is gondoskodnunk kell. Azt, hogy egy lap tartalma megváltozott-e vagy sem, a laptáblában lévő úgynevezett dirty – piszkos – bit mondja meg).

Létezik egy algoritmus, ami tökéletes megoldást nyújt erre a nehézségre. Működése roppant egyszerű: minden lapkerethez egy számlálót rendel, ami azt mondja meg, hogy hány utasítás elteltével lesz szükség a betöltött lapra. Amikor egy lapot el kell távolítani, értelemszerűen azt dobjuk ki, amelyikre a legeslegkésőbb lesz szükségünk (tehát amelyiknek a számlálója a legnagyobb értéket veszi fel), így elodázhatjuk a következő lapcserélés időpontját. Ennek a módszernek csupán egyetlen hibája van: megvalósíthatatlan.

Egy futó programról ugyanis nem tudjuk előre megmondani, hogy melyik lapra hány utasítás múlva lesz szüksége. Kivéve azt az esetet, amikor előtte már egyszer lefuttattuk, és egy különleges alkalmazás segítségével kiértékeljük a program által kért laphivatkozásokat. Az így nyert adatok alapján már hatékony lapcserélési eljárás építhető össze. Ezzel azonban nem jutunk messzire, mivel egy másik program esetén valószínűleg nem lenne hatékony – teljesen balsikerű megoldást valósítana meg.

Nézzünk meg néhány olyan algoritmust, aminek a való életben is hasznát tudjuk venni. Ezek közül a legegyszerűbb a FIFO (First-In, First-Out), ahol mindig a legrégebben betöltött lap helyére tesszük be az újat. Nagyon nagy hátránya, hogy nincs tekintettel rá, hogy az adott lapot eddig hányszor használták, így az a lap is eldobásra kerül egyszer, amire eddig a legtöbbször volt szükség – a közeljövőben nagy valószínűséggel ismét be kell majd tölteni.

Gyakran egész hatékonyan használható az NRU (Not Recently Used, nem mostanában használt) eljárás, ahol a lapokat négy osztályba soroljuk a következők szerint:

1. nem hivatkozott és nem is módosított,
2. nem hivatkozott, de módosított,
3. hivatkozott, de nem módosított,
4. hivatkozott és módosított is.

A kidobandó lapot mindig véletlenül választja ki a legkisebb sorszámmal rendelkező, ám nem üres csoportból.

A fent említett hatékony algoritmus egyik legjobb megközelítése azonban az LRU (Last Recently Used, legutóbb használt), ahol mindig azt a lapot dobjuk el, amelyikre a legrégebben volt szükség. Ez már egy létező, a való életben is használható módszer, csak a szolgáltatásait nem kínálja olcsón, ugyanis mindig nyilvántartásba kell venni, hogy melyik lapnak kell a legközelebbi laphibánál távoznia. Ezt láncolt lista formában célszerű tárolni, ami minden egyes lapcserélésnél karbantartásra szorul – ez pedig még eszközszintű megoldás esetén is időigényes feladat.

A munkahalmazmodell

Azt, hogy egy memóriakezelő mennyire hatékonyan működik, nemcsak az általa használt lapcserélési algoritmus határozza meg, hanem sok más tényező is. Ezek közé tartozik például az is, hogy mennyire tudjuk elkerülni egy adott program vergődését. Akkor mondjuk, hogy egy program vergődik, ha a futása során túl sok laphibát okoz. Talán felesleges megemlíteni, hogy ez milyen kellemetlen jelenség, hiszen a program végrehajtási sebessége jelentősen csökken, mivel a processzornak a lapok ki-bepakolásával kell az ideje nagy részét eltöltenie.

A hagyományos lapozási rendszerekben egy lemezen lévő lap csak úgy kerülhet a memóriába, ha valakinek az adott pillanatban szüksége van rá (ez a módszer az úgynevezett igény szerinti lapozás). Amikor egy programot elindítunk, akkor semelyik általa használt lap sincs a memóriában, tehát futásának első felében vergődni fog. Ez az állapot egészen addig tart, amíg a program munkahalmaza (az általa használt lapok halmaza) be nem töltődik.

Ez még nem is lenne akkora szerencsétlenség, de többfeladatos rendszer esetében számolnunk kell azzal is, hogy ha nincs elég memória az összes futó alkalmazás munkahalmazának befogadására, akkor néhány blokkolt program lapjait ki kell raknunk a lemezre. Amikor ezek a programok ismét futni kezdenek, megint nagyszámú laphibát fognak okozni, mivel az egész munkahalmazuk ki lett helyezve a központi tárból.

A fenti nehézség a munkahalmazmodell segítségével oldható meg, ez nem jelent mást, mint hogy figyelemmel kísérjük a programok munkahalmazát, és még a program indítása előtt betöltjük őket a memóriába. (Ezt az operációs rendszer általában úgy csinálja, hogy megnézi, milyen lapokra hivatkozik egy-egy program. Ezeket mind beteszi az adott folyamat munkahalmazába. Ha azonban egy lapra egy meghatározott időn belül nem érkezik kérés, akkor kidobja onnan).

Szakaszolás és osztott könyvtárak

Mint láthattuk, a lapozás nagyon hasznos művelet, mivel megszabadítja a programozót az olyan hosszú és unalmas tevékenységektől, mint például a program rétegekre bontása. Emellett a lapozás kényelmesen használható, mivel az egész memóriát egy összefüggő egydimenziós tömbként látjuk, és nem kell olyan „apróságokkal” foglalkoznunk, mint hogy az adott memóriaszelet éppen a lemezen vagy a központi tárban található-e.

Akadnak azonban olyan pillanatok, amikor kifejezetten jól jönne, ha nem egy, hanem több, egymástól teljesen független címtartománnyal gazdálkodhatnánk. A legjellemzőbb eset az, amikor több táblázatot kell a memóriában a program futása közben beérkező adatokból felépítenünk. A probléma gyökerei nyilván oda vezetnek vissza, hogy a program indulásának pillanatában halvány sejtelmünk sem lehet arról, hogy melyik táblázat pontosan mennyi helyet fog felemészteni a memóriából.

Az ilyen esetekre a szakirodalomban példaként legtöbbször a fordítóprogramokat hozzák fel. A forráskód végigpásztázása során különböző táblákat kell felépítenie a memóriában, amelyek közül az egyikbe kerülnek majd az állandók, egy másikba a változók és így tovább.

Ha ezt színtisztán a lapozás segítségével szeretnénk megoldani, akkor a memóriát annyi egyenlő részre kell felosztanunk, ahány táblázatra szükségünk van. Na igen, de tegyük fel, hogy a programozó elképesztően sok állandóval dolgozik lefordítani kívánt programjában, így a fordítás során előbb-utóbb megtelik az állandóknak fenntartott memóriaterület. Ilyenkor a fordítóprogramnak komoly nehézséggel kell szembenéznie, mivel nem növelheti tovább az állandók táblájának méretét, hiszen ütközne egy másik táblával, például a változókéval.

Ebben az esetben a fordítóprogramnak két választása van – az első az egyszerűbb megoldás: pánikba esik, kiír egy hibaüzenetet, és befejezi a futását. Ez szükségtelen, mert lehet, hogy más táblák még rengeteg hellyel rendelkeznek, így második lehetőségképp el lehetne „rabolni” az üres helyet azoktól, akiknek még van.

Ez ígéretes, csak megint ott tartunk, hogy a programozónak saját kezűleg kell egy olyan hosszadalmas és fárasztó munkát elvégeznie, amelyet igazán rábízhatnánk az operációs rendszerre is. Így valamilyen más módszer kell, amivel a virtuálismemória-kezelés mellett a változó méretű adatszerkezetekkel is könnyedén boldogulhatunk. Ez pedig a szakaszolás.

A szakaszolás egy olyan virtuálismemória-kezelő eljárás, ami lehetővé teszi, hogy több egymástól különálló címteret (szakaszt) hozhassunk létre.

Ha ez netán egy kicsit tisztázatlan lenne, úgy képzeljük el, mintha a lapozás az egydimenziós, a szakaszolás pedig a kétdimenziós memória megfelelője lenne. Lapozás esetében egy adott bájt eléréséhez elegendő volt egy „koordináta” megadása, míg a szakaszolásnál már kettő szükséges. Az első azt mondja meg, hogy melyik szakaszban (címtéren) kell keresgélnünk a második koordinátaként megadott címet. Ezt az utóbbi értéket gyakran ofszetnek is nevezik.

Ám a szakaszolás nem csupán a memória több címtérre való szabdalása. A szakaszok mérete ugyanis a program futása során folyamatosan változhat: nőhet is, csökkenhet is. Mindezt úgy, hogy nem „ütközhet” a többi szakasszal, mivel minden szakasz saját címtérrel bír. Egy szakasz nem nőhet a végtelenségig, ugyanis mindegyiknek van egy legnagyobb mérete, amelyet nem léphet túl. Mindezek ellenére ez az érték általában elég ahhoz, hogy a szakasz a program futása alatt ne teljen be (csak ha a program helyesen működik).

A szakaszolást azonban már nem tudjuk elrejteni a programozó elől, miként azt a lapozás során tettük. A szakaszokkal a felhasználóknak is foglalkozniuk kell, és egyfajta logikai egységekként kezelik. Azt is meghatározhatják, hogy melyik szakaszba mi kerüljön. Így külön szakaszba költözhetnek az állandók, a változók, a verem és az eljárások is. Egy fontos kitétel létezik: a különböző típusú szakaszokat nem szabad keverni, tehát amelyekbe állandókat helyezünk, azokban valóban csak állandók legyenek stb.

A változó méretű adattáblák egyszerűbb kezelhetőségén túl a szakaszolás jó megoldást nyújt az adatok és eljárások könnyed megosztására is. Az utóbbi főleg a grafikus környezetben futó alkalmazások számára létfontosságú, amiben, ha a megosztást nem lehetne megvalósítani, akkor még a legkisebb programhoz is hihetetlen méretű függvénykönyvárat kellene hozzáfűznünk. A szakaszolás segítségével azonban a hívható eljárásokat egyszerűen csak be kell pakolni egy szakaszba, majd azt meg kell osztani a grafikus könyvtár és a felhasználói alkalmazások között. Ez az osztott könyvtár fogalma.

A szakaszolás megvalósítása sok tekintetben hasonlít a lapozáshoz, azzal a különbséggel, hogy míg minden lap ugyanakkora méretű, addig egy szakasz hossza bármekkora lehet. Ezért számolnunk kell a memória töredezésével (fragmentation). Ez azt jelenti, hogyha egy szakaszt kiveszünk a memóriából, akkor ott egy vele megegyező méretű lyuk keletkezik. Értelemszerűen oda csak ugyanakkora vagy annál kisebb méretű szakasz kerülhet. Előbb-utóbb szembesülnünk kell azzal a nehézséggel, hogy egy nagyobb szakaszt nem tudunk anélkül betenni a memóriába, hogy egy másikat el ne

távolítanánk, mivel nincs akkora méretű lyuk. Ugyanakkor elférne a memóriában, mert van annyi szabadhelyünk, csak épp mindenfelé elszórva. Ilyen esetben a tömörítést kell alkalmaznunk, azaz az összes szakaszt olyan módon kell eltolnunk lefelé, hogy ezek a lyukak megszűnjenek, és a felső részen egy összefüggő szabad terület alakuljon ki.
Ha a szakaszok mérete a központi memóriához képest túl nagy, akkor egyszerre csak kevés szakaszt tudunk ott tartani, ami hatékonyságbeli romláshoz vezet. Ennél is rosszabb, ha egy szakasz nagyobb, mint maga a központi tár. Ezért a legtöbb rendszer támogatja a szakaszolás és a lapozás keresztezéséből létrejövő módszert: a lapozott szakaszolást. Ez tulajdonképpen semmi mást nem takar, mint hogy a szakaszokat egyenlő méretű lapokra osztjuk fel. Ennek köszönhetően nem kell mindig az egész szakaszt betölteni a memóriába, csupán azokat a lapjait, amikre az adott pillanatban szükség van.

Az Intel Pentium

Végezetül szeretnék pár szót szólni a Pentium processzorok virtuális memóriakezeléséről, és arról, hogy miképp is zajlik a gyakorlatban.
A Pentium támogatja a lapozott szakaszolást. Egyszerre több mint 16 ezer szakaszunk lehet és akár 4 GB méretűre is megnőhetnek. A szakaszok nyilvántartására két táblázat szolgál: a helyi leírótábla (Local Descriptor Table – LDT) és a globális leírótábla (Global Descriptor Table – GDT). Az utóbbiból csak egy van, ebben többnyire az operációs rendszer által használt szakaszok foglalnak helyet. LDT-ből már több is akad, minden programhoz tartozik egy, amiben saját szakaszaikat raktározzák.
A szakaszok tárolásán kívül fontos még a védelem kérdése, ennek segítségével az operációs rendszer szakaszait megvédhetjük a felhasználói programoktól, mindezt eszközszinten téve.
A Pentium processzorban négy védelmi szint van megadva, ezek közül a nulladik a legerősebb és a harmadik a leggyengébb. Mind a programoknak, mind a szakaszoknak van egy szintszámuk. Minden program írhatja és olvashatja a vele egy szinten lévő szakaszokat, de az alacsonyabb szintű szakaszok esetén ez a művelet nem megengedett. Más szinten lévő szakaszok eljárásai viszont szabadon hívhatók.
Ezeket a védelmi szinteket a rendszer jól ki tudja használni. Például a 0-s szintre bepakolhatjuk a rendszermag szakaszait, az elsőre a rendszerhívásokat, a másodikra az osztott könyvtárakat és a legutolsó szintet pedig a felhasználói programok foglalhatnák el. Így például megoldható, hogy a felhasználói alkalmazások szabadon hívhassák az osztott könyvtárak eljárásait, viszont azok módosítására nem lenne lehetőségük.
Vajon az operációs rendszerek (például a Linux) ki is használják a védelmi szintek által nyújtott szolgáltatásokat? Ez kiderül a következő részből.

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.

Beszédbe elegyedünk a konyhafőnökkel...

Bemutatjuk a geektalkd és a Shadowlands Forum csevegőkiszolgálót, valamint a Jabbert, ezt a kitűnő azonnali üzenetküldő ügyfélprogramot.

Minden remekül fest, François. Minden asztalon gyertya, borospoharak, munkaállomások és DSL-kábelek – úgy tűnik, elkészültünk. A ma esti fogásokkal az Elektronikus Fórumok előtt tisztelgünk. Ezek mindegyikének az a célkitűzése, hogy közelebb hozza az embereket egymáshoz, mégha csak látszólagosan is. Talán valamilyen romantikus bort kellene felszolgálnunk – az 1997-es Alsatian Gewurztraminer jónak tűnik a ma esti alkalomra. Nem értesz velem egyet, mon ami? Nos, romantikáról talán tényleg túlzás beszélnünk, de gondold csak meg: az emberi érintkezés, még ha pusztán a hálózaton keresztül valósul is meg, a kapcsolattartásról szól, ez pedig bizonyos közelséget is jelent, még ha nem is vonzódunk a partnerhez, akivel kapcsolatban állunk. Quoi? Mindez nagyon egyszerű, François. A Világháló létezésének is ez a gondolat áll a középpontjában: az emberek kapcsolatot keresnek egymással. Ez történik egy weboldal esetén, ami valamilyen terméket szeretne eladni, de ugyanez a helyzet, amikor a munkatársak elektronikus levelet váltanak egymással. Ha a Linuxszal kotyvasztunk, számos különböző módon élhetünk ezekkel a kapcsolatteremtési lehetőségekkel. Megérkeztek a vendégeink – bonsoir, mes amis! Isten hozott titeket Chez Marcelnél, az ízletes Linux-fogások és a legkitűnőbb borok otthonában. Az asztalok terítve, kérem, foglaljatok helyet! A Világhálón az elektronikus kapcsolatteremtési lehetőségeknek számos módját fedezhetjük fel, mes amis, a webportáloktól kezdve a belső hálózatokon (intranet) át egészen a csevegőszobáig, sőt az azonnali üzenetváltásig (instant messaging). A rendelkezésünkre álló eszközök lehetőséget kínálnak a már folyó vitákhoz, eszmecserékhez történő csatlakozásra, de akár a saját rendszerünk is otthont adhat ilyennek. A Linux tökéletes felületet nyújt kiszolgálók működtetésére, és mindez nem is kíván nagy erőfeszítést. Bizonyság erre ***Michael Plump*** `geektalkd` nevű Perl nyelven írt és ***David Elkins*** hibajavításaival tökéletesedett csevegőkiszolgáló programja. Mivel egyszerű felépítésű, nyílt forrású programról van szó, könnyen módosítható, továbbfejleszthető, ami tovább növeli az értékét. Támogatja a csatornákat, a magánbeszélgetéseket és még sok egyebet.
A `geektalkd` telepítéséhez és futtatásához szükségünk lesz a `Net::Ident` Perl-modul telepítésére. Ezt legegyszerűbben a Perl CPAN üzemmódjának a segítségével tehetjük meg.
A teljes telepítési folyamat az alábbi paranccsal végezhető el:

```
perl -MCPAN -e "install Net::Ident"
```

Amennyiben ez az első alkalom, hogy a Perl CPAN parancsértelmezőjét használjuk valaminek a telepítésére, válaszolnunk kell néhány, a rendszer által feltett kérdésre. A legtöbb esetben a felkínált választ elfogadhatjuk, de akad néhány kivétel is, például a tükrözés kiválasztása esetében. Attól függően, hogy akadt-e már korábban dolgunk a CPAN-blokkal, a folyamat időigénye egy és néhány perc között változhat.
Ha ezzel készen vagyunk, csomagoljuk ki és fordítsuk le a programot:

```
tar -xzvf geektalkd-1.23.tar.gz
cd geektalkd-1.23
perl install
```

A kiszolgáló futtatásához elég a program nevét begépelni, ezzel indítjuk el a háttérben a folyamatot. Ha naplófájlt is szeretnénk működtetni, használhatjuk a `-l` kapcsolót, a `-p` beállítással pedig a használt kaput adhatjuk meg. A `geektalkd` alapértelmezésként a 41000-es kaput használja:

```
/usr/local/sbin/geektalkd -l /var/log/gtd_log
➥-p 41001
```

Ha már fut a kiszolgáló, az ügyfélprogramnak nem kell mást tennie, mint `telnet`-tel a kérdéses kaput megcéloznia:

```
telnet your_site.dom 41001
```

A kiszolgáló egy üzenettel nyugtázza a kapcsolódást.
A `/help` begépelésével a kiadható parancsokról kapunk listát.
A különösen könnyen használható csevegőkiszolgálók másik példája a ***Jeremy Monin***, ***Dan Chin*** és ***Matt Lochansky*** megalkotta Shadowlands Forum. A program különösen sokoldalú csevegőszoba, néhány kitűnő (esetenként csak a szórakoztatást szolgáló) képességgel. A felhasználók például beállíthatják távollétüzenetüket, üresjárati időkorlátjukat, írhatnak makrókat, engedélyezhetik a levéllellenőrzést (helyi felhasználó esetén) és így tovább. A rendszer felügyelője beállíthat csak bizonyos felhasználókra érvényes alapértékeket, megadva, hogy ki tartozik abba a bizonyos körbe, továbbá témaköröket adhat meg és egyéb beállításokat is tehet. A Shadowlands Forum kipróbálásához töltsük le a programot a ➲ http://www.shadowlands.org/forum címről, és csomagoljuk ki egy helyi könyvtárba:

```
tar -xzvf slforum-1.9.6.1.tar.gz
cd slforum-1.9.6
```

Továbblépés előtt az `slprefs.h` állományban kell néhány változtatást végrehajtani. Ez a fájl tartalmazza az olyan helyi beállításokat, mint az `slforum` saját könyvtára és fájljainak a helye. Ha a rendszerünk árnyékjelszavakat használ, a `#define USE_SHADOW` bejegyzést hatástalanítanunk kell (az alapértelmezett beállítás a PAM). A rendszerüzenetek szerkesztésére is itt nyílik lehetőségünk, de természetesen az eredeti üzeneteket is megtarthatjuk. Ha mindennel elégedettek vagyunk, a következőképpen fordíthatjuk le a programot:

```
make -f Makefile-linux
```

Ebben az esetben nincs `make install`, az `slforum` futtatható fájlt egyszerűen másoljuk egy alkalmas könyvtárba. PAM használata esetén az ***slforum.pam*** fájlt a PAM könyvtárába

történő átmásolására van szükség (`cp slforum.pam /etc/pam.d/slforum`). Itt az idő, hogy életre keltsük új csevegőszobánkat:

```
./slforum -ag marcel +aa /path_to/chat.log
```

A `-ag` kapcsoló a rendszergazdák csoportját (***admin group***) azonosítja a kiszolgáló számára, míg a `+aa` beállítás bekapcsolja a vendégek önműködő fogadását (***guest auto-accept feature***). Ami a parancs végén látható naplófájlt illeti, az tetszőleges lehet. A kapcsolók listázását a `-h` kapcsolóval kérhetjük a programtól. A fórumhoz való csatlakozás bármilyen `telnet` ügyféllel lehetséges, méghozzá a 7777 kapun keresztül:

```
telnet your_site.dom 7777
```

Ilyen módon akár vendégként, akár rendszergazdaként belépheténk. A vendégektől azonosítót (vagy nevet) kér a rendszer, a rendszergazdáknak jelszót kell beírniuk. Bejelentkezés után a `!help` begépelésével az elérhető parancsokról bármikor listát kérhetünk. Sőt rendelkezésünkre áll a bűvös 8-as golyó (magic 8 ball, egy népszerű jövendőmondó játék – a fordító), amitől az élet összetettebb kérdéseire is választ várhatunk:

!8 ball Should I serve Chenin Blanc with on-line fora?
FORUM: The 8 ball has advised marcel:
To Answer Would Dishonor the 8 ball
(Vagyis: marcel> !8 ball Szolgáljak fel Chenin Blancot* a hálózati fórum mellé?
FORUM: A 8 ball tanácsa marcel számára:
A válasz sértené a 8 ball önérzetét.
(* A Chenin Blanc egy híres dél-afrikai borfajta – *a fordító.*)
Nos, mes amis, lehet, hogy ez a 8-as golyó nem is olyan hasznos szolgáltatás.

Mai utolsó fogásunk egy azonnali üzenetküldő ügyfélprogram. Sokan megkérdezhetik, hogy mi a különbség a már bemutatott csevegőkiszolgálók és az azonnali üzenetküldés között. Valójában a kettő elég hasonló. A különbség az, hogy az eddig megvizsgált programok mindegyike azt igényelte, hogy egy központi helyhez csatlakozzunk. Csak akkor dönthetjük el valakiről, hogy éppen elérhető-e, ha bejelentkezünk a kiszolgálóra. Az olyan azonnali üzenetküldő protokollok, mint a Jabber (➲ http://www.jabber.org) valós idejű eszmecserét tesznek lehetővé barátunkkal vagy rokonunkkal (akiket az (Instant Messaging IM, azonnali üzenetküldés terminológia buddynak – haver, pajtás – nevez), ugyanúgy, mint az előző példákban láthattátok. A különbség az, hogy az IM-ügyfelek egymással állnak kapcsolatban, így tudhatjuk, hogy a „haverunk” éppen vonalban van-e. A Jabber egyébként nyílt, nem szabadalmazott XML alapú IM-protokoll, ami választási lehetőséget jelent a többi szolgáltatás mellett. Valójában a Jabber nyílt mivolta azt jelenti, hogy saját kiszolgálót futtathatunk, ami a magánhálózatok szempontjából igen előnyös.

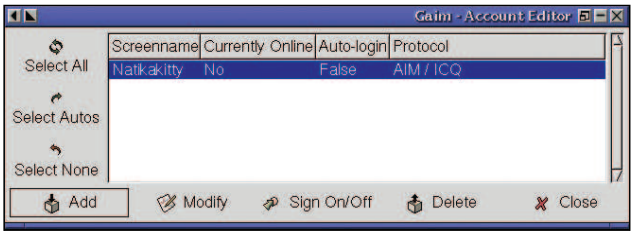
1. kép A Gaim első indítása

Szinte biztos, hogy a rendszerünkön már fut egy kitűnő Jabber-képes IM-ügyfél, amelyet Gaimnek neveznek. Eredetileg ***Mark Spencer*** írta, jelenleg ***Rob Flynn*** foglalkozik karbantartásával (természetesen az elhivatott kódolók és foltokat alkotók segítségével). A Gaim segítségével most egy Jabber-azonosítót fogunk létrehozni, de a program ugyanígy lehetőséget nyújt arra, hogy az AOL, Yahoo, a Microsoft-féle MSN vagy egyéb protokollt használó ügyfelekhez csatlakozzunk. A protokollok szempontjából ezt a sokoldalúságot a Gaim bővítmények segítségével éri el, és válik ezáltal az elérhető egyik legrugalmasabb IM-ügyfélprogrammá.

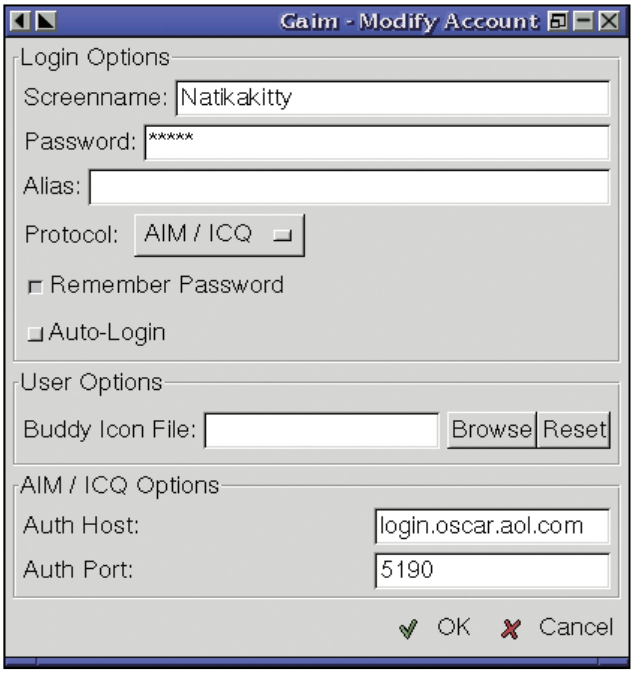
Ha mégsem lenne meg a gépünkön a Gaim, a legfrissebb forráskódot a ➲ http://gaim.sourceforge.net címről tölthetjük le. A csomag fordítása a jól bevált ötlépéses kicsomagoló–fordító módszerrel történik:

```
tar -xzvf gaim-0.59.6.tar.gz
cd gaim-0.59.6
./configure
make
su -c "make install"
```

Akár a menüből, akár a `gaim &` paranccsal parancssorból elindítva a programot, a főablakot látjuk megjelenni (*1. kép*).



2. kép Jabber-fiók létrehozása vagy módosítása



3. kép Ilyen egyszerű egy új felhasználói fiók felvétele vagy beállítása

Első alkalommal egy egyszerű ablak jelenik meg, ami a képernyő nevét (***Screen Name***) és a jelszót (***Password***) várja. A képernyőnév mezőjében a ***<New User>*** (új felhasználó) szavakat fogjuk látni, amiből kiderül, hogy még egyetlen felhasználói fiókot sem állítottunk be. Nyilvánvaló, hogy

mielőtt használatba vennénk a Gaimet, legalább egy fiókra szükségünk lesz, non?
Ha a szöveges mezők alá tekintünk, hat gombot láthatunk a következő feliratokkal: ***Quit*** (kilépés), ***Accounts*** (felhasználói fiókok), ***Signon*** (bejelentkezés), ***About*** (névjegy), ***Options*** (beállítások) és ***Plugins*** (bővítmények). Alapértelmezésben a Gaim ICQ-azonosítót is elfogad, ha már rendelkezünk ilyennel. Ahhoz, hogy a Jabber protokollt használni tudjuk, előtte be kell töltenünk a hozzá tartozó bővítményt, mert ez alapértelmezésben nem aktív. Ez rendkívül egyszerű. Kattintsunk a ***Plugins*** gombra, mire megjelenik a Gaim bővítményeket kezelő ablaka. Kattintsunk a ***Load*** (betöltés) gombra és a felbukkanó listából válasszuk ki a kívánt protokollt. Jabber esetén a ***libjabber.so*** fájlra, majd az ***OK***-ra kell kattintani. Ugyanígy választható ki a Yahoo protokol (***libyahoo.so***) vagy az MSN csevegő protokollja (***libmsn.so***). Ha a Jabber protokollját (vagy valamelyik másik, nekünk tetszőt) betöltöttük, a ***Close*** (bezár) gombbal térhetünk vissza. A bővítmények ablakának bezárása után kattintsunk az ***Accounts*** gombra. Erre megjelenik az ***Account Editor*** (fiókszerkesztő) ablaka, egyelőre üresen. Kattintsunk az ***Add*** (hozzáadás) gombra, ennek hatására megjelenik a ***Modify Account*** (fiók módosítása) feliratú ablak. Az ablak közepe táján egy legördülő menüt látunk ***Protocol*** felirattal. Alapértelmezettként itt az AIM/ICQ a kiválasztott – változtassuk Jabberre és figyeljük meg, ahogy az ablak a Jabberhez szükséges beállításoknak megfelelően megváltozik.
Írjuk be a képernyőnk nevét (***Screenname***), a jelszót (***Password***) és a másodnevet (***Alias***), és kattintsunk a ***Register with server*** (a kiszolgálóval döntő bejegyzés) rádiógombra. Ha azt szeretnénk, hogy a Jabberbe való bejelentkezés minden alkalommal önműködően hajtódjon végre, ha az ügyfélprogramot elindítjuk, kattintsunk rá az ***Auto-Login*** feliratú rádiógombra is. Ha mindennel végeztünk, kattintsunk az ***OK*** gombra. A fiókbeállító (***Account Editor***) ablakban megjelenik az új felhasználói fiók (*3. kép*).
A bejelentkezést megejthetjük innen is a ***Sign On/Off*** gombra való kattintással, vagy a ***Close*** gombbal bezárjuk az ablakot, és a Gaim főablakából választjuk ki a megfelelő műveletet. Első bejelentkezésünkkor egy üdvözlő üzenetet kapunk a Jabber.org kiszolgálótól. Ezt bezárhatjuk, vagy további útbaigazításért felkereshetjük a kiszolgáló honlapját (ahogy arról a szöveg is tájékoztat).
Most, hogy elkészültünk saját Jabber azonnali üzenetküldő fiókunkkal, társakra van szükségünk, akikkel kapcsolatba léphetünk. Számos hálózati beszélgetésbe bekapcsolódhatunk a menüsor ***File/Join a Chat*** pontjának kiválasztásával. Ugyanezt a műveletet a CTRL-C billentyűkombinációval is elérhetjük. Adjuk barátainkat a buddylistához az ***Add A Buddy*** menüpont választásával.
Miután ismerőseinket hozzáadtuk a listához, egy tájékoztató üzenetet kapnak, hogy a listánkhoz szeretnénk őket fűzni. Amikor megpillantják az előugró ablakot, az ***Accept*** (jóváhagy) gombra kattinthatnak, s ezután már kezdődhet is a beszélgetés. Ennek a jóváhagyásnak a kapcsolat mindkét oldalán meg kell történnie. Ő elfogad téged, ezt követően te is elfogadod őt. Úgy gondoljunk erre, mint a boldogító „igen” kimondására, csak éppen egy véletlenszerűbb, mondhatni látszólagos viszony esetében. François, az ilyen barátságok pezsgőért kiáltanak, nem gondolod?
Mialatt François felhozza a pezsgőt, befejezzük a Gaim vizsgálatát. Ha ezek a jóváhagyások lejátszódtak, „haverjaink” megjelennek a listánkon. A nevek mellett egy ikon jelzi, hogy az adott partner éppen vonalban van-e. Ha igen, kettőskattintás a néven, és már indulhat is a csevely. Ennyi az egész.
A Gaim szintén gazdag beállítási lehetőségeket kínál, amelyekről a ***Tools*** (eszközök) majd a ***Preferences*** (felhasználói beállítások) pontot kiválasztva kaphatunk listát. Az egyik kedvencem (a ***Conversations*** beszélgetések pont alatt) az a lehetőség, hogy minden folyó beszélgetés egyetlen, fülekkel tagolt ablakra helyezhető. Ha egyszerre sok eszmecserét folytatunk, erre biztosan szükségünk lesz.
Á, François, jó, hogy visszaértél! Kérlek, tölts a barátainknak! Tartok tőle, hogy a záróra már megint a nyakunkon van. Igyunk egymás egészségére a következő havi találkozásunkig! A votre santé! Bon appétit!

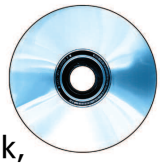
Linux Journal 2003. március, 107. szám

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

KAPCSOLÓDÓ CÍMEK

Gaim-honlap ➲ http://gaim.sourceforge.net
geektalkd ➲ http://www.geektalk.org
Jabber ➲ http://www.jabber.org
Shadowlands Forum ➲ http://www.shadowlands.org
Marcel borlapja ➲ http://www.marcelgagne.com/wine.html

Fordítás az Apache Ant segítségével



A make típusú programokat felváltani kívánó Ant egyre jobban terjeszkedik, ezért nem árt egy kicsit jobban is megismerkedni vele…

Aki fordított már programot Unix/Linux alapú rendszereken, az biztosan találkozott a `make` programmal. A `make` programnak számos előnye mellett rengeteg hátránya is akad. Ilyen például az, hogy nagymértékben függ az adott héjtól (shell), és biztosan másokkal is előfordult már, hogy a TAB lemaradt a megfelelő sor elejéről. E hiányosságok megszüntetése végett került sor az Ant kifejlesztésére.
Az Ant egy olyan Java alapú fordítóeszköz (build tool), amelynek segítségével operációs rendszertől függetlenül fordíthatunk programokat, a leggyakrabban Java-alkalmazásokat. Ezt az Ant úgy tudja megtenni, hogy a nagymértékben héjparancsokra épülő `Makefile` helyett az XML-írásmódot követő ***build.xml*** fájlt használja. Természetesen a ***build.xml*** lehetőséget teremt arra, hogy az operációs rendszertől függő utasítások hajtódjanak végre, de erről később még lesz szó.

Telepítés és beállítás

Debian alatt a telepítése nagyon egyszerű:

```
# apt-get install ant
```

Cikkünk írásának pillanatában az Ant legfrissebb üzembiztos változata az 1.5.1-es. A Woodyban ugyan még az 1.4.1-es változat található, de a Sarge-ban már az 1.5.1-es szerepel. Akiknek nem Debian rendszere van, azok a telepítés két módját is választhatják:

- vagy a ➲ http://www.apache.org/dist/ant/source/ címről töltik le a legfrissebb forrást és a szükséges ***jar***-fájlokat,
- vagy pedig a ➲ http://www.apache.org/dist/ant/binaries/ címről szedik le a futtatható állományokat.

A második az egyszerűbb megoldás, és ha csak pár alkalommal szeretnénk használni az Antot, vagy még soha nem próbáltuk ki a Javát vagy Java alapú rendszereket, akkor első lépésben érdemes a binárist feltenni. A letöltéshez használhatjuk a fent említett oldalakat vagy a tükröoldalakat.
Töltsük le a binárist, és rendszergazdaként csomagoljuk ki a ***/usr/local***-ba:

```
/usr/local # wget -O -
➥http://www.apache.org/dist/ant/binaries/
➥apache-ant-1.5.1-bin.tar.bz2 | tar jxf -
```

Az eredmény egy ***jakarta-ant-1.5.1*** nevű könyvtár lesz a ***/usr/local***-ban, ami a következő fontosabb könyvtárakat tartalmazza:

- ***bin***: a parancsfájlokat tartalmazza.
- ***lib***: az Ant működéséhez szükséges ***jar***-fájlok jegyzéke.
- ***doc***: a leírásokat foglalja magában.

Ha Debian alatt `apt-get`-tel telepítettük az Ant programot, akkor a könyvtárszerkezet a következőképpen alakul:

- ***/usr/share/ant/bin***: a parancsfájlok helye.
- ***/usr/share/ant/lib***: a működéshez szükséges ***jar***-fájlok helye.
- ***/usr/share/doc/ant***: a leírások helye.

Azért fontos, hogy az adott rendszeren hol helyezkednek el a fájlok, mert egyes Java-alkalmazások telepítésénél meg kell adni az `ANT_HOME`-ot, azaz azt, hogy az adott gépen hol található meg az Ant program. Ha erre sokszor szükségünk lesz, érdemes az `ANT_HOME` környezeti változót a ***/etc/environment***-ben megadni:

```
~# echo ANT_HOME=/usr/local/
➥jakarta-ant-1.5.1 >> /etc/environment
```

Ha nem szeretnénk minden egyes Ant-utasításnál a teljes elérési útvonalat megadni, a `PATH` környezeti változót a következőképpen módosítsuk:

```
export PATH=$PATH:$ANT_HOME/bin
```

Debian alatt ezt a lépést akár ki is lehet hagyni, hiszen az `ant` parancsra a ***/usr/bin***-be Ant néven egy közvetett hivatkozás (symlink) kerül, s ezt a jegyzéket a `PATH` rendszerint tartalmazza.

Projekt

Az Ant a ***build.xml*** fájlt használja a projektek (projects) leírására, ezért az XML alapszintű ismerete szükséges ahhoz, hogy az Ant programot használni tudjuk. Azok számára, akik láttak már HTML-t, az XML formája nagyon ismerős lesz.
Az XML egy olyan szövegfájl, ami adatokat ír le, jelen esetben majd a fordítás értékeit. Minden ***build.xml*** fájlnak tartalmaznia kell egy projektet (project) és legalább egy célt (target). A célok a projekten belül helyezkednek el. Ha az Ant indításakor nem adjuk meg, hogy melyik célt vagy célokat hajtsa végre, akkor a projekten belül az alapértelmezettként (default) megadott cél kerül végrehajtásra.
A projektnek három értéke lehet:

- `name`: a projekt nevét adja meg (nem kötelező).
- `default`: az alapértelmezett célt adja meg (kötelező).
- `basedir`: a munkakönyvtárat adja meg (nem kötelező megadni és később felülbírálható). *1. listánk* egy folyamat megadását szemlélteti.

```
<project name="Elso" default="kiir"
basedir=".">
      <target name="kiir">
            <echo message="Elso build.xml
                  ➥probalkozas" />
      </target>
</project>
```

```
<project name="Masodik" default="kiir"
        ➥basedir=".">
    <target name="kiir" />
    <target name="beallit" />
    <target name="fordit"
          ➥depends="kiir,beallit"/>
</project>
```

```
<project name="Harmadik" default="keszit"
        ➥basedir="/tmp">
    <property name="celkonyvtar"
          ➥value="build" />
    <target name="kiir">
        <echo message="A ${celkonyvtar}
              ➥elkeszitese.."/>
    </target>

    <target name="keszit" depends="kiir">
        <mkdir dir="${celkonyvtar}" />
    </target>
</project>
```

A célok

A célok akciókat írnak le, azaz megmondják, hogy az egyes cél eléréséhez milyen műveleteket kell végrehajtani.
A célok között függőségeket is megadhatunk. Ha például azt szeretnénk, hogy a ***.class*** fájlok majd rendre a ***build*** könyvárban legyenek, akkor nyilván először a könyvtárat kell elkészíteni, azután a ***.class*** fájlok is átkerülhetnek a megadott könyvtárba. Ha a *2. listában* látható ***fordit*** nevű célt szeretnénk végrehajtani, akkor először nem az fog lefutni, hanem a ***kiir***, majd a ***beallit***, és végül a ***fordit***.
A célnak öt értéke lehet:

- `name`: a cél nevét adja meg (kötelező).
- `depends`: a függőségeket adja meg, ha több más céltól is függ, akkor vesszővel kell elválasztani őket egymástól.
- `if`: lehetőség nyílik a feltételes fordításra, azaz az adott cél csak akkor kerül végrehajtásra, ha egy adott tulajdonság (property) be van állítva.
- `unless`: az if ellentéte, azaz ha be van állítva, akkor nem kerül végrehajtásra.
- `description`: arra ad lehetőséget, hogy az adott cél szerepét, esetleg működését leírjuk.

A feladatok

Egy feladatot (task) a következőképpen lehet megadni:

```
<feladatnev attributum1="ertek1"
➥attributum2="ertek2" ... />
```

A feladatok megadása biztosítja a legnagyobb szabadságot. Rengeteg beépített feladat (`java`, `javac`, `jar`, `war`, `echo` stb.) létezik, de saját magunk is meghatározhatunk ilyeneket. A ***feladatnev***-nek megfelelő program kerül majd végrehajtásra, azokkal az értékekkel, amelyeket a fenti formában adhatunk meg. Például az `<mkdir dir="build"/>` bejegyzés hatására egy `mkdir build` parancs fog végrehajtódni. A feladatok pontos meghatározását a ➲ http://ant.apache.org/manual/coretasklist.html oldalon találhatjuk meg.

```
<project name="Negyedik" default="kiir"
        ➥basedir=".">
    <property environment="env" />
    <target name="kiir">
        <echo message="A TERM
        ➥kornyezeti valtozo erteke
        ➥= ${env.TERM}"/>
    </target>
</project>
```

```
public class Film
{
    private String Title;
    private String Genre;
    private String Director;

    public void Film()
    {
        Title = Genre =
            ➥Director =
            ➥null;
    }
    public void setTitle(String Title)
    { this.Title = Title; }
    public void setGenre(String Genre)
    { this.Genre = Genre; }
    public void setDirector(String
                        ➥Director)
    { this.Director = Director; }

    public String getTitle()
    { return Title; }
    public String getGenre()
    { return Genre; }
    public String getDirector()
    { return Director; }

}
```

A tulajdonságok (properties)

A tulajdonság is egy feladat (task), ami lehetőséget teremt arra, hogy egy folyamat tulajdonságait beállítsuk. E tulajdonságokra a folyamaton belül a `${tulajdonsagnev}` formában lehet hivatkozni. A *3. listában* megadott feladat már egy kicsit összetettebb. A ***celkonyvtar*** tulajdonságra kétszer is létezik hivatkozás. Ezenkívül érdemes a `basedir` értéket is szemügyre venni. Ha az Ant programot a 3. listának megfelelően futtatjuk le, létrejön a ***/tmp/build*** könyvtár. Öt lehetőségünk van arra, hogy egy tulajdonságot beállítsunk:

- a `name` és a hozzá tartozó `value` megadása.
- a `name` és a hozzá tartozó `refid` megadása.
- a `file` értéket használva a hivatkozott fájl nevével (itt a `name`-et nem szabad megadni).
- a `resource` értéknek és a hivatkozott fájl nevének a megadása (itt szintén nem szerepelhet a `name`).
- az `environment` érték használatával és egy előtag (prefix) megadásával (lásd a *4. listát*).

```
<project name="Mintaprogram"
➥default="fordit" basedir=".">
      <description>
            Minta program ant mukodesere.
      </description>

      <!-- tulajdonsagok beallitasa -->

      <property name="src" location="src"
/>
      <property name="build"
                  ➥location="build" />
      <property environment="env" />

      <target name="kiir">
            <echo message="Korny ezeti
                        ➥valtozok:" />
            <echo message="JAVA_HOME:
                  ➥${env.JAVA_HOME}" />
            <echo message="ANT_HOME:
                  ➥${env.ANT_HOME}" />
      </target>


      <target name="keszit">
      <!-- A szukseges konytar(ak)
➥elkeszitese (ha meg nem leteztek elotte) -->

            <mkdir dir="${build}" />
      </target>

      <target name="fordit"
                  ➥depends="kiir,keszit">
      <!-- a ./src ben levo java forrasok
      ➥leforditasa a ./build -be -->

            <javac srcdir="${src}"
                  ➥destdir="${build}" />
      </target>


      <target name="clean">
        <!-- a java binarisok torlese -->

            <delete dir="${build}"/>
      </target>

</project>
```

Az első lépések…

Most már rendelkezünk az alapismeretek nagy részével – itt az ideje, hogy a tettek mezejére lépjünk. A `-f` kapcsoló feladata ugyanaz, mint a `make`-é: kiválaszthatjuk vele a leírófájlt. Ha nem adjuk meg, akkor a munkakönyvtárban lévő ***build.xml*** fájlt fogja használni. Az `ant` parancs után egy vagy több cél megadására is lehetőség nyílik. Ha egyet sem adunk meg, akkor értelemszerűen az alapértelmezett cél fog végrehajtódni (ezért kötelező minden esetben ennek az értéknek a megadása). A 3. listában megadott folyamatot mentsük egy XML-fájlba, adjunk neki egy tetszőleges nevet (***harmadik.xml***), és nézzük meg a következő két utasítást:

```
~$ ant -f harmadik.xml

~$ ant -f harmadik.xml kiir
```

Az első esetben nem adtunk meg célt, ezért az alapértelmezett cél hajtódik végre. Ennek az a feladata, kiírjon egy üzenetet, ezt követően készítsen egy könyvtárat. Ezzel szemben a második parancskiadás már egy célt tartalmaz, mégpedig a ***kiir*** nevűt. De ennek nem feladata a könyvtárkészítés, ezért csak a kiírás történik meg.
Az Ant nem arra való, hogy a képernyőre írogassunk vagy könyvtárakat készítsünk vele. Ezek csak segédszolgáltatások. Most lássunk egy példát egy Java-program fordítására: adott a ***Program.java*** (*5. lista*) és ***Film.java*** (6. lista) Java-forrás. Ezek a pillanatnyi könyvtár ***src*** alkönyvtárában találhatók. A következő lépéseket kell végrehajtani, ha le szeretnénk őket fordítani:

```
~$ mkdir build
~$ ls
build src
~$ javac -d ./build ./src/Film.java
~$ javac -classpath ./build -d ./build
./src/Program.java
~$ cd build
~$ java Program
```

Ez még nem annyira nagy gond, de ha már több osztály van, és a ***jar***-fájlok is szükségesek, akkor a fenti módszer nem sok jót ígér. Ugyanakkor, ha a *7. listán* megadott ***build.xml*** fájlt használjuk, elegendő ennyit beírni:

```
~$ ant
~$ cd build
~$ java Program
```

A *7. listában* szereplő fájlban az újdonság a `<javac srcdir="${src}" destdir="${build}" />` folyamat(task), mely az összes ***src*** könyvtárban lévő Java-forrást lefordítja a ***build*** könyvtárba (itt is jól látszanak a hivatkozások a tulajdonságokra).
A fenti műveleteket `make` programmal is meg tudnánk csinálni, viszont `make`-kel nem tudnánk az ***src*** könyvtárat és a ***build.xml***-t egy másik operációs rendszerre átvinni, és ott módosítás nélkül lefordítani ezt az egyszerű programot.
Az Ant remek tulajdonságainak köszönhetően (Java-alapok, XML használata) alapja lehet a jelenkor és a jövő operációsrendszer-független alkalmazásfejlesztéseinek.

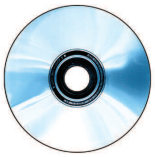
A listák megtalálhatóak a 46. CD Magazin/Ant könyvtárában.

Fabók Zsolt (fabokzs@zib.iit.uni-miskolc.hu)
A Miskolci Egyetem Informatika szakos hallgatója és demonstrátora. A ZiB linuxos társaság oszlopos tagja. Szabadidejében osztott rendszerekkel foglalkozik.

KAPCSOLÓDÓ CÍMEK

- http://ant.apache.org/
- http://java.sun.com/xml

Alkalmazásfejlesztés a Zope segítségével



Webes alkalmazások varázsütésre, néhány egérkattintással, avagy hogyan fogjuk munkára Zope-ot, a nagy varázslót.

Talán még sosem volt ennyire fontos a sebesség és a hatékonyság, mint napjainkban. A mában születő ötletekből hozzuk létre a jövő world wide webjét. Rohanó világunkban a jövő csak egy karnyújtásnyira van, és minden ötletből gazdája a lehető leghamarabb valami működő, kézzelfogható dolgot szeretne látni. A lehetőségek adottak, csak a megfelelő eszközt kell kiválasztanunk.
Cikksorozatunkban egy webes alkalmazásfejlesztő eszközzel ismerkedünk meg, a Zope-pal. A Zope valójában egy HTTP-kiszolgáló, ennek köszönhetően mind a beállításait, mind adataink kezelését böngészőn keresztül végezhetjük.
A Zope tudja mindazt, amit a jelenkor igényei megkövetelnek:

- Teljes mértékben objektumokra épül, így biztosítva a már megalkotott elemek újra-felhasználhatóságát.
- Egyszerű felhasználói és programozói felülettel bír, így a kezdő felhasználók is könnyedén elsajátíthatják a használatát.
- Beépített tartalomkezelő rendszert nyújt, amivel azok is könnyen boldogulnak, akik egyébként nem járatosak a programozás világában.
- Beépített jogosultsági rendszert tartalmaz, így pontosan beállítható, hogy a rendszer felhasználóinak milyen objektumokhoz legyen hozzáférése.
- A különböző változatok kezelése is beépített tulajdonság, így egyetlen fájlból egyetlen könyvtárban akár több változatot is tárolhatunk anélkül, hogy a különböző változatok bármilyen módon zavarnák egymást.
- A Zope naplózó rendszerének köszönhetően a Zope felhasználói felületén végzett műveletek szinte kivétel nélkül visszafejthetők.
- A Zope teljes egészében nyílt forrású, így bárki a saját elképzelései szerint módosíthatja az alkalmazást.
- A Zope-hoz írt programok felületfüggetlenek, azaz bármilyen rendszeren működőképesek, ahol a Python és a Zope fut.

A legelső pontot, miszerint a Zope objektumokra épül, külön is érdemes kihangsúlyozni, hiszen a Zope a PHP-vel és az ASP-vel ellentétben teljesen objektumközpontú. A Zope erőteljesen támaszkodik a Python programnyelv lehetőségeire, ugyanis maga a Zope is Pythonban készült – eltekintve néhány kisebb résztől, amit a nagyobb teljesítmény érdekében C-ben írtak meg. A Pythonon kívül a Zope lehetőséget teremt a Perl használatára is, így ki-ki maga döntheti el, hogy számára melyik programozási nyelv a rokonszenvesebb.
A tartalom kialakítására és rugalmassá tételére is több lehetőség kínálkozik, sőt ez az a terület, amelyben a Zope igazi ereje lakozik. Ez a két eszköz a DTML (Document Template Markup Language) és a ZPT (Zope Page Templates) fogalmat takarja.
A két eszköz nagyjából hasonló, mindkettővel HTML-fájljainkba szúrhatjuk be a Zope-nak szóló parancsokat. A DTML a tartalom kezelésére sokkal szélesebb körű szolgáltatásokat nyújt, a használata azonban manapság már inkább kerülendő, több szempontból is. Elsőként a DTML túlságosan bonyolult felületek kialakítását is lehetővé teszi, ami már programozásnak minősül, és így a logika elveszti egyszerűségét. A második, ami talán a legfontosabb: ha egy DTML-kóddal teleszurkált HTML-fájlt betöltünk a kedvenc weblapszerkesztőnkbe, akkor nagy valószínűséggel a szerkesztő szépen eltávolítja a parancsainkat, mivel nem ismeri őket.
Ezt az utóbbi gondot a ZPT úgy oldotta meg, hogy nem úgynevezett tagokat, hanem a tagok mögé másik névterülethez tartozó kapcsolókat szúr be, amit aztán a weblapszerkesztő programok (legalábbis az újabbak) nem bántanak.
De lássuk csak, miről is van szó! Alább látható egy DTML-parancs:

```
<p><dtml-var title></p>
```

Míg ennek ZPT-s megfelelője:

```
<p tal:content="template/title">Az oldal
➥c me</p>
```

Majd a későbbiekben látni fogjuk, hogy az egyes szerkezetek pontosan mit is jelentenek.

A Zope telepítése

A Zope-ot a ➲ http://www.zope.org ***Download*** menüpontja alól tölthetjük le, ami mindig a legfrissebb változatot tartalmazza. Ha azonban egy a Linux-terjesztésünkhöz hangolt változatra van szükségünk, jó eséllyel kereshetjük a Zope-ot a terjesztésünkhöz mellékelt CD-ken is, mivel elterjedtségének és népszerűségének köszönhetően ma már a nagyobb linuxos cégek is mellékelik.
Ha Debian GNU/Linuxot használunk, elég a következő parancsot begépelnünk:

```
apt-get install zope
```

A telepítést követően a Zope-ot a Zope könyvtárában kiadott `./start` paranccsal indíthatjuk el. Debian esetében az alkalmazást nem szükséges külön elindítani, mivel az telepítés után magától elindul. Ha valamilyen okból kifolyólag mégsem indult volna el, használhatjuk a következő parancsot is:

```
/etc/init.d/zope start
```

Ezt követően a kedvenc böngészőnket is indítsuk el, és csatlakozzunk a Zope felhasználói felületéhez a következő cím begépelésével:

http://localhost:8080/

Ha a Zope-ot Linux-terjesztésünk CD-jéről telepítettük, elképzelhető, hogy nem a 8080-as kapun kell csatlakozni, hanem egy másikon, de ez a CD-hez adott leírásból kiderül. Debian eseté-

ben például a 9673-as kapun. Ha se a 8080-as, se a 9673-as kapun nem jártunk sikerrel, és a dokumentáció sem tér ki erre a dologra, akkor a `netstat -ntl` parancs kiadásával megnézhetjük, hogy milyen kapuk vannak nyitva a gépünkön. Ha találunk egyet a 8–9000 körüli tartományban, akkor nagy valószínűséggel az lesz a mi Zope-unk.
Amennyiben a Zope-ot nem önálló webkiszolgálóként szeretnénk futtatni, hanem mondjuk Apache vagy IIS alatt, a Zope arra is lehetőséget ad.
Ha a böngészőnkben sikerült behoznunk a Zope kezdőoldalát, kattintsunk az itt megjelenő ***Zope Management Interface*** hivatkozásra, vagy a böngésző címsorába gépeljük be a következő címet:

http://localhost:8080/manage

A belépéshez először egy úgynevezett ***emergency***, vagyis vészhelyzet-felhasználót kell létrehoznunk. Ezt csak rendszergazdaként tudjuk megtenni, ugyanis a Zope első használata során csakis ezzel a felhasználóval tudunk belépni. Ehhez először be kell lépnünk abba a könyvtárba, ahova a Zope-ot telepítettük, majd ki kell adni a következő parancsot:

```
# python zpasswd.py access
```

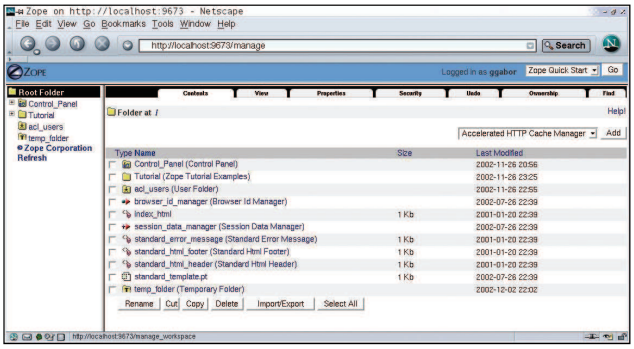
Debianban is hasonlóképpen működik:

```
# /usr/sbin/zope-zpasswd /usr/lib/zope/access
```

A vészhelyzet-felhasználó neve bármi lehet, de általában ***emergency***-nek vagy ***admin***-nak szokás nevezni. Ez a felhasználó egyébként különös szerzet, mivel nevezhető nagyhatalmúnak, ugyanakkor gyengének is. Gyenge, mivel az ***emergency*** felhasználóval csak más felhasználók beállításait módosíthatjuk, egyéb elemekhez nem férünk hozzá; de ugyanennek következtében nevezhető nagyhatalmúnak is, mivel a Zope-ban lévő összes felhasználót módosíthatjuk, újakat vehetünk fel vagy régieket törölhetünk.
Ennek okán ha a vészhelyzet-felhasználóval sikeresen bejelentkeztünk, hozzunk létre magunknak egy általános felhasználót. Lépjünk be az ***acl_users*** könyvtárba, és az ***Add*** gombra kattintva ***Manager*** és ***Owner*** jogosultságokkal vegyük fel a felhasználónkat. Ezt követően a vészhelyzet-felhasználónkkal lépjünk ki, és az újonnan létrehozott felhasználóval jelentkezzünk vissza. Innentől kezdve már bármilyen objektumot létrehozhatunk, felhasználókat módosíthatunk, vagy bármit megtehetünk, ami csak az eszünkbe jut (lásd a *képet*).

A Zope használata

A Zope-ba bejelentkezve, ahogy azt már észrevehettük, egy fájlkezelőszerű ablakot kapunk. És valóban, az egész nem más, mint egy fájlkezelő. A bal oldali keretben lehet kiválasztani a pillanatnyi könyvtárat, a jobb oldalon pedig a könyvtárban állíthatunk a megfelelő tulajdonságokon.
A mappákban új objektumokat felvenni a jobboldali keret felső sarkában található legördülő listával és az ***Add*** gombbal lehet. Objektumnak minősülnek a könyvtárak is, így amennyiben mondjuk a saját könyvtárban, tehát a ***Root Folder***-ben egy új könyvtárat szeretnénk felvenni, a legördülő listából jelöljük ki a ***Folder*** elemet, és kattintsunk az ***Add*** gombra. ***ID***-nek, vagyis új objektumazonosítónak adjuk meg a létrehozandó könyvtár nevét, ami legyen mondjuk „teszt”; ***title***-nek, vagyis az elem címének pedig adjuk a sokat sejtető „Teszt mappa” címet.



A Zope induláskor

Ha ezzel megvagyunk, kattintsunk megint csak az ***Add*** gombra, és csodáljuk meg a létrejött könyvtárat. Minden Zope-objektum rendelkezik egy azonosítóval, amivel hivatkozhatunk az objektumra. Felfoghatjuk ezt úgy is, mint egy változónevet. Ezenkívül minden objektumnak van egy ***title*** tulajdonsága, ami az objektumhoz tartozó címet hivatott megadni. ID-nek csak olyan nevet választhatunk, ami semmilyen különleges karaktert nem tartalmaz, mivel ezzel az azonosítóval kell majd a későbbiekben a böngészőből is hivatkozni, hogy az objektumot elérhessük.
Itt érdemes megjegyezni, hogy minden egyes ablakban a jobb felső sarokban találunk egy ***Help!*** hivatkozást, ami mindig az adott tartalomhoz kapcsolódó súgót jeleníti meg. Így ha egy ablakban nem vagyunk biztosak az egyes tulajdonságok jelentésében, vagy nem tudjuk, mi a teendő, nyugodtan forduljunk a ***Help!***-hez segítségért.
Az objektumokra a jobb oldali mezőben található lenti eszköztárral lehetünk hatással. E gombok segítségével nevezhetünk át, mozgathatunk, vagy törölhetünk objektumokat. A ***cut***/***copy***/***paste*** gombok teljesen úgy működnek, mint ahogyan azt bármely szövegszerkesztőben vagy fájlkezelőben megszokhattuk. Jelöljünk ki egy elemet, mondjuk a ***standard_html_header*** nevű objektumot, tehát kattintsunk a név mellett található jelölőnégyzetbe, ezt követően pedig kattintsunk a ***Copy*** gombra. Ezután lépjünk be a teszt mappánkba, és kattintsunk a ***Paste*** gombra. Mint ezután láthatjuk, a másolás sikeresen megtörtént, ahogyan arra számítottunk is.
Az ***Import***/***Export*** gomb arra szolgál, hogy a Zope-ba objektumokat emeljünk be (import), illetve onnan emeljünk ki (export). Ennek azért van nagy jelentősége, mert a Zope objektumai nem a fájlrendszerben helyezkednek el, tehát Linux fájlrendszerünkből közvetlenül nem férünk hozzájuk, hanem egy úgynevezett konténerfájlban találhatók, amihez csak a Zope-on keresztül juthatunk hozzá. Emiatt létezik egy külön ZopeFTP, amit a 8021-es kapun keresztül szokás elérni. Ide belépve a Zope fájlrendszerében tallózhatunk, és másolhatunk fájlokat, akárcsak ha bármilyen más FTP-kiszolgálóra léptünk volna be.
Az ***Import*** és ***Export*** menük ***.zexp***-fájlokat kezelnek. Ha a későbbiekben a Zope-unkhoz valamilyen alkalmazást töltünk le, azt is ilyen ***.zexp***-fájlból beemelve tudjuk a Zope-ból elérhetővé tenni.
A jobb oldali keretben a felső sarokban további füleket találhatunk. Eddig csak a ***Contents*** fül tartalmával ismerkedtünk meg, amiből az objektumok közvetlenül érhetők el, azonban létezik még néhány érdekes fül. Az egyik ezek közül a ***View***, amire kattintva az adott könyvtár feldolgozott tartalma látható, vagyis az, amit később jó esetben a Webről is el lehet érni. Az itt látható

tartalomra igaz, hogy amennyiben közvetlenül nem adunk meg objektumnevet, először egy ***index_html*** fájlt keres a könyvtárban, és ha talál ilyet, annak a tartalmát jeleníti meg.
Számunkra érdekes lehet még az ***Undo*** fül, itt az eddigi műveleteink listáját fedezhetjük fel. Bármelyiket kijelölve és az ***Undo*** gombra kattintva visszavonhatjuk őket. Megjegyzendő, hogy csak olyan eseményt „csinálhatunk vissza”, aminek a Zope szempontjából van értelme. Ha tehát egy könyvtárba bemásolunk egy fájlt, majd töröljük a könyvtárat, akkor a kérdéses fájlt csak úgy állíthatjuk vissza, ha a mappa törlését is visszavonjuk.

Ismerkedés a Zope objektumaival

Mint már említettem, a Zope objektumokkal dolgozik, emiatt minden, amivel a Zope alatt dolgozunk, objektumok formájában érhető el. Objektumok a könyvtárak, a fájlok és a HTML-oldalak is. A jobb oldali keretben az ***Add*** gombhoz tartozó legördülő listát tallózva további elemeket láthatunk. Ilyen elem például a ***Temporary Folder***, amiből máris találunk egy példányt a saját könyvtárban. A ***temp_folder***-ben található elemeket a rendszer a memóriájában tárolja, így ha a Zope-ot újraindítjuk, a tartalma elvész. Mivel az itt található objektumok a memóriában tárolódnak, figyeljünk arra, hogy ezt a könyvtárat ne töltsük meg túlságosan, mert ilyen módon rendszerünk összes memóriáját elfogyaszthatjuk.
Az ***DTML Document*** nem más, mint amiről már korábban is szó volt. A ***Page Template*** pedig a ZPT-t rejti, amivel ZPT-oldalakat hozhatunk létre.
A DTML- vagy ZPT-dokumentum létrehozásánál ugyanúgy meg kell adni az ***ID***-t és ***Title***-t, mint minden objektum esetében, és emellett arra is lehetőségünk van, hogy az újonnan létrejövő objektumhoz egy már létező fájlt töltsünk fel, ha a ***File*** mezőt kitöltjük. Amennyiben nem töltünk fel fájlt, úgy a Zope az objektumot egy alapértelmezett tartalommal tölti fel, amit nyugodtan átszerkeszthetünk.

Sablonoldalak

A „page template”-ek oly módon könnyítik meg a programozók és a grafikusok munkáját, hogy az így készített oldalak szinte bármilyen honlapszerkesztőben szabadon szerkeszthetők, anélkül, hogy az oldalakba épített logika sérülne. Mint már említettem, ez annak köszönhető, hogy a ZPT-vezérlők nem külön HTML-címkékként épülnek be az oldalba, amit a honlapszerkesztő program alkalomadtán szó nélkül kivághat, hanem a létező HTML-címkékhez kapcsolódnak, és így sértetlenül megúsznak egy alapos külalak-átváltoztatást is, feltéve, hogy a szerkesztőprogram tiszteletben tartja a ZPT kapcsolóit. A mai szerkesztőprogramok többsége szerencsére már megfelelően viselkedik ilyen helyzetekben is.
A ZPT a TAL-t, vagyis a ***Template Attribute Language***-et használja az oldalszerkezet kialakításakor. A TAL a HTML-címkékbe épül be, a `tal` névtér alá, vagyis a ZPT-nek szánt utasításainkat a `tal:` kulcsszóval kell kezdenünk:

```
<title tal:content="here/title">Az oldal
➥c me</title>
```

A névtér egy XML-fogalom, amivel esetünkben a `tal:` különbözteti meg a ZPT-nek szóló kapcsolókat a HTML kapcsolóitól, így minden olyan honlapszerkesztő, ami felismeri az XML-t, a ZPT-utasításokat sem fogja bántani.
A fenti példában a ZPT „Az oldal címe” szöveg helyére a `title` változóhoz kapcsolódó tartalmat fogja beszúrni. A ZPT

1. lista Egy új sablon

```
<html>
  <head>
    <title tal:content="template/title">Az
                    ➥oldal c me</title>
  </head>
  <body>

    <h2><span tal:replace="here/title_or_id">
            ➥A tartalom c me vagy
            ➥azonos t ja</span>

        <span tal:condition="template/title"
             ➥tal:replace="template/title">
             ➥A tartalom c me, ha meg van
             ➥adva</span></h2>

    Ez egy sablon <em tal:content=
    ➥"template/id">sablon azonos t </em>.
  </body>
</html>
```

utasításaival nemcsak a HTML-címkék közé szúrhatunk be szöveget, de a teljes címkeszerkezet lecserélésére is lehetőségünk van. Az így végrehajtódó utasítások minden alkalommal önműködően mennek végbe, valahányszor csak valaki megtekinti az oldalt.
Az egyetlen dolog, amire a ZPT nem képes, az a bonyolult utasítássorozatok vagy a végrehajtódás folyamatát szabályozó műveletek kezelése, mint amilyenek például a függvények, vagy az osztályok. Az efféle műveleteket kizárták a ZPT-ből, mivel az ilyen bonyolultságú műveletekre a Zope más lehetőségeket biztosít.

A sablonok gyakorlati alkalmazása

Első lépésként felhasználónkkal be kell jelentkezni a Zope kezelői felületére (a ZMI-re), majd a gyökérkönyvtárban létre kell hoznunk egy ***zpt-proba*** nevű könyvtárat, a könyvtárhoz pedig hozzá kell adunk egy `Page template` típusú elemet, majd pedig, ha a böngészőnk nem lép önmagától tovább, az ***Add*** gombra kell kattintanunk. Ezt követően beállíthatjuk ZPT-elemünk azonosítóját (ID), amivel az elemet majd a Weben keresztül el lehet érni, emiatt a nevét is ennek megfelelően kell megválasztani, tehát nem tartalmazhat sem ékezetet, sem egyéb különleges karaktert. Az első oldalunk neve legyen ***elso_oldal***. A fájl mező kitöltésére csak akkor van szükség, ha egy létező fájlt szeretnénk feltölteni a Zope könyvtárszerkezetébe. Esetünkben egy új sablont szeretnénk szerkeszteni, így kattintsunk az ***Add*** gombra, majd pedig a létrejött ***elso_oldal*** nevén kattintva megtekinthetjük a fájlunk tartalmát. Ha az ***Add*** gomb helyett előzőleg az ***Add and Edit*** gombon kattintottunk, azonnal a szerkesztőoldalra juthattunk.
Mint látható, a Zope nagyon elkényeztet minket: nem kell egy teljesen új sablont szerkesztenünk, hanem felajánl egy kezdetleges sablont, amit tetszés szerint átszerkeszthetünk vagy akár teljesen kitörölhetünk, esetleg kibővíthetünk (*1. lista*).
A `tal:content` kapcsoló a HTML-címkék közötti részt, vagyis a tartalmat cseréli le, tehát ha megadjuk a következőt:

```
<p tal:content="template/title">Az oldal
➥c me</p>
```

akkor a ZPT ezt a következőképpen fordítja le a böngészőnek:

```
<p>Az oldal c me, amit a title vÆltoz ban
➥beÆll tottunk.</p>
```

Mint látható, „Az oldal címe” szöveg teljes egészében elveszik. Ez a szöveg csak arra jó, hogy a programozónak vagy a grafikusnak támpontot nyújtson, hogy eredetileg milyen tartalmat szántak az adott helyre, de a felhasználó szempontjából lényegtelen, mivel a ZPT teljes egészében eltünteti.
Más a helyzet a `tal:replace` kapcsolóval, ami az egész címkeszerkezetet lecseréli. Lássuk, hogyan néz ki egy ezt a kapcsolót használó HTML-sor:

```
<p tal:replace="template/title">Az oldal
➥c me</p>
```

A böngésző ebből csupán ennyit fog látni:
Az oldal címe, amit a `title` változóban beállítottunk.
Tehát a `<p>` címke, és a `</p>` zárócímke teljes egészében eltűnik, és a helyére a `tal:replace`-nek megadott változó tartalma kerül. A `tal:replace` esetén a `<p>` helyett ajánlott a `<span>` HTML-címke használata, mivel ez önmagában semmilyen kinézetet befolyásoló jelentéssel nem bír. Bár a ZPT és a böngésző számára ez teljesen mindegy, a programozó és a grafikus számára, aki a honlapszerkesztőben látja az oldalt, zavaró lehet egy látszólag felesleges új bekezdést jelölő címke, míg a `<span>` használata esetén a szerkesztendő szöveg a szerkesztőprogramban úgy jelenik meg, ahogyan a felhasználó böngészőjében fog.
A `tal:content` kapcsoló egy egyszerű feltételt takar, és az azt követő `tal:replace` kapcsoló csak akkor fog kiértékelődni, ha a feltételben megadott változó tartalma adott.

A sablonokról egy kicsit bővebben

A sablonok segítségével lehetőség van egyszerű vezérlési szerkezetek, például ismétlések kezelésére is, amelyek alkalmazására a 2. listában (46. CD Magazin/Zope) láthatunk példát. A korábban tanultakat alapul véve talán nem okoz nehézséget a ZPT-szerkezet megértése, de azért lássuk csak, miről is van szó. A `tal:repeat` kapcsolóban megadott második elem, a `container/objectValues` a sablon könyvtárában található fájlok hivatkozásait tartalmazza, amiken a `repeat` utasítás sorban végiglépdel, és minden egyes menetben egy változót ad értékül az első elemként megadott `elem` változónak. Ilyen módon az `elem` változón keresztül lehetőségünk nyílik a könyvtárban található minden egyes állomány tulajdonságainak lekérdezésére. Az `elem/getId` tulajdonság az éppen feldolgozás alatt lévő elem azonosítóját jelenti, míg az `elem/title` a feldolgozás alatt álló elem címkéjét. A `meta_type` tulajdonság tartalmazza az adott állomány típusát.
A `repeat/elem/number` tulajdonság az adott ciklusra vonatkozik. Azt mutatja, hogy az adott elem hányadik a sorban. Mivel a `tal:repeat` kapcsolók egymásba ágyazhatók, ha a kereséshez tartozó tulajdonságokhoz szeretnénk hozzáférni, meg kell adnunk azt is, hogy pontosan melyik `tal:repeat` ciklusra gondolunk, ezt a `repeat/` után megadott ciklusazonosító változóval tehetjük meg. Ilyen módon a ZPT már pontosan tudni fogja, hogy a `repeat/elem/number` tulajdonság keresésekor melyik ciklus adataival kell visszatérnie. A `number` helyett használhatjuk az `index` tulajdonságot is, ami annyival különbözik, hogy a számozást nem egytől, hanem nullától kezdi. Használható tulajdonságok még a `letter` és a `Letter` elemek, amik az egyes sorokat betűvel jelölik, a `letter` esetében kisbetűvel, a `Letter` esetében nagybetűvel.
A táblázatban a negyedik oszlop megadásánál a következő sor látható:

```
<img tal:attributes="src elem/icon">
```

Itt a `tal:attributes` használatára láthatunk példát. Az `attributes` kapcsoló annyit tesz, hogy az adott HTML-címke (vagyis esetünkben az `img`) egy kapcsolóját módosítja. A módosítandó kapcsoló neve az `attributes` kapcsolóban megadott szöveg első része (src), majd ezt követi a változó, amit a kapcsolónak értékül akarunk adni, vagyis az `elem/icon` változó, ami az adott típushoz tartozó ikon elérési útját tartalmazza. A `tal:attributes` tehát a HTML-címke egy kapcsolójának a tartalmát módosítja, vagyha az még nem létezik, akkor létre is hozza. A fenti sor hatására a ZPT a következő HTML-kódot küldi a böngészőnek:

```
<img src="ikon elØrØsi œtja">
```

Összegzés

Az előző oldalakon a Zope-pal ismerkedhettünk meg. Egy rendszer, ami különbözik mindentől, amivel eddig programozóként vagy webes fejlesztőként találkozhattunk. Ötvözi a hatékony programozás és a gyors, kattintgatásos honlapszerkesztés előnyeit – mindezt meglepően logikus formában tálalva, hogy még a kezdő felhasználók is elsőre megértsék működését, ugyanakkor a szakavatott felhasználók számára is minden lehetőség biztosított legyen.
Ha kedvet kaptál ahhoz, hogy alaposabban megismerkedj a Zope-pal, a ***Kapcsolódó címek*** részben található oldalakon elindulva kellőképpen elmélyítheted az ismereteidet.
A cikksorozat következő részeiben a Zope néhány alkalmazásával ismerkedünk meg. Szó lesz a Plone-ról, a felhasználók által legjobbnak ítélt tartalomkezelő rendszerről, és kitérünk a Zope komolyabb lehetőségeire is.

Gludovátz Gábor (ggabor@sopron.hu)
Hobbija a programozás és a bringázás. Szereti a Perl és a Python hatékonyságát, a PHP egyszerűségét, a Java logikáját, és a C++ kibogozhatatlan szépségét. Szabadidejében úszik vagy bringán a szülővárosát körülölelő dombokon tör a legmagasabb csúcsok felé. Honlapja a ➲ http://gludo.sopron.hu címen érhető el.

KAPCSOLÓDÓ CÍMEK

A Zope hivatalos oldala ➲ http://www.zope.org
A Zope leírása és a „Zope könyv”
➲ http://www.zope.org/Documentation
A DevShed Zope-bevezetője
➲ http://www.devshed.com/Server_Side/Zope/Intro
ZopeLabs – példák, hivatkozások, hírek
➲ http://www.zopelabs.com

Sablonelvű fejlesztés a Smarty segítségével (1. rész)

Ismerkedjünk meg a Smarty sablonkezelő motorral, amelynek a segítségével egyszerűbbé tehető a küzdelem a szebb programkódért és a karbantarthatóbb küllemért.

Hajdanán, amikor először megláttam és megszerettem a PHP nyelvet, az szólt mellette az egyik fő érvként, hogy mindenféle furfangoskodás és ***cgi-bin*** könyvtárak nélkül, a HTML-oldalakba ágyazgatva alkothattam kisebb-nagyobb programokat. Most, évekkel később egyre inkább kódjaim olvashatóságának (és ezzel karbantarthatóságának), javításának sziszifuszi feladatával küszködöm. Egyre inkább zavar az a sok HTML-kód, ami miatt a program logikája nehezen áttekinthető.
Régóta kacsintgattam már a sablonnyelvek felé, de a megszokás is nagy úr. Csakhogy elég volt egy kíváncsi délután, és a próbálkozásból izgalmas felfedezés kerekedett ki. Ezt a felfokozott állapotot nemcsak a tiszta programkód idézte elő, és nem is az, hogy innentől kezdve akár a „látom, amit készítek" szerkesztő segédeszközöket is használhatnám (ezektől valahogy még mindig ódzkodom) – a legnagyobb élményem abból a felfedezésből fakadt, hogy valóban milyen határozottan el tudom választani a száraz logikát és a külcsínt egymástól. Innentől kezdve nincs szükségem a programkódban kacifántos feltételes elágazásokra csupán azért, hogy a megjelenített listám minden páros sora színben eltérhessen a páratlanaktól, és nem kell több tucat programsort az egyik PHP-fájlomból a másikba mozgatnom azért, hogy a bal felső sarokban figyelő legördülő listámat a jobb alsó sarokba száműzhessem. Ez a módszer azzal hódította meg teljesen a szívemet, hogy az elkészült színes, szagos weboldal nyomtatható változatának előállításához már csak egy kis sablonmunka szükséges, a prógramkód pedig maradhat ugyanaz. E sablonrendszer nélkül ez a feladat a teljes oldalmegjelenítő programkód klónozását jelentené, amelyben ezután már lebutíthatnám a megjelenést, mindezt a kevesebb felesleges nyomtatófesték fogyasztásának szent ügyéért.
Lelkendezésből tán ennyi elég is, lássuk, mi is rejtőzik az igencsak felvágós Smarty név mögött!

Előfordítás

A Smarty használata előnyösebb számos másik sablonkezelő rendszerénél, mivel a megjelenés minőségéért felelős sablonállomány értelmezése nem történik meg minden oldallekéréskor. Ehelyett PHP-parancsállományokat készít belőlük, és mindaddig ezeket használja, amíg nem módosítunk az adott sablonon. Egy oldal akár több ilyen darabból is összeállítható (fejléc, test, lábléc sablonok külön állományban), ilyenkor csak az adott oldaldarabka kerül újra feldolgozásra, ha változott benne valami. E trükk segítségével megmarad a programkód és a megjelenés elkülönítése, ennek ellenére sincs semmiféle sebességbeli hátránya a dolognak.

Sokrétű tudás

A Smartynak az a tulajdonsága, hogy sok mindent tud, egyesek szerint előny, mások szerint hátrány. A bonyolultabb sablonnyelvet ellenzők fő érve az, hogy teljesen felesleges egy újabb összetett nyelvet bevezetni a Smartyval. Igaz, ami igaz:

➲ http://smarty.php.net

a Smarty sablonnyelvén már-már kisebb programokat is írni lehet. Véleményem szerint ez nem okoz gondot, hiszen ezeket a tulajdonságokat nem szükséges kihasználni. A Smarty egyszerű sablonnyelvként is használható, anélkül, hogy a nem használt elemek bármiféle lassulást okoznának a rendszerre nézve. Mivel a Smarty előfordítást végez, a nyelvi lehetőségek számossága nem befolyásolja a futás sebességét. Ennek a nyelvi sokrétűségnek az előnye, hogy általa rugalmasabban osztható szét a munka az arculatért felelős honlapszerkesztő és a programozó között, ismerve azok képességeit.

Telepítés és használatbavétel

Nos, ha már eddig eljutottunk, ismerkedjünk meg cikkünk tárgyával. Ehhez először is szükség lesz egy frissen letöltött ***tar.gz*** állományra. Nosza, vegyük elő a kedvenc böngészőnket, és látogassunk el a ➲ http://smarty.php.net weboldalra.
A ***Download*** menüpont mögött rejtező oldalról pillanatok alatt beszerezhető a legfrissebb változat – a csomag mai viszonylatok szerint igen kicsiny (160 KB) méretének köszönhetően. (A Smarty a CD-mellékleten a Magazin/PHP könyvtárban is megtalálható.) A letöltött állomány izgatott kicsomagolása után tekintsük meg, mi is van a „dobozban".
Sok minden helyet kapott a csomagban, de a munkánkhoz nem mindenre lesz közülük szükségünk. Ami viszont elengedhetetlen, az a három ***class.php***-re végződő állomány ***Smarty.class.php***, ***SmartyCompiler.class.php*** és ***Config_File.class.php***, valamint a ***debug.tpl*** és a teljes ***plugins*** könyvtár. Ezeket érdemes olyan helyre pakolni, ami benne van valamelyik, az `include_path` beállítások szerint kijelölt könyvtárban. A világháló számára is látható területen belül, ahol a Smartyval meghajtott munkánk életre fog kelni, három könyvtárra lesz szükség: ***templates***, ***templates_c*** és a ***configs***. Ne feledjünk meggyőződni róla, hogy a ***templates_c*** könyvtárban webkiszolgálónk is képes dolgozni, azaz állományokat, könyvtárakat létrehozni, illetve törölni. Itt fogja ugyanis a sablonállományokból előállítani a HTML-tartalommal kevert PHP-kódot.
Ezeknek az állományoknak és könyvtáraknak a helyére

1. lista
A smarty_proba.tpl csupán egy aprócska sablonállomány

```
<html>
<head>
 <title>{$nev} - hozzavalok</title>
 <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-2">
</head>
<body>

<p>
<b>{$nev} (hozzavalok):</b>

<ul>
{section name=i loop=$hozzavalok}
     <li>
     {$hozzavalok[i].mennyiseg}
     <i>{$hozzavalok[i].egyseg}</i>
     <b>{$hozzavalok[i].nev}</b>
{/section}
</ul>
</p>

<p>
Ha kØrdØsed van:
{mailto address=$email encode="javascript"}
</p>

</body>
</html>
```

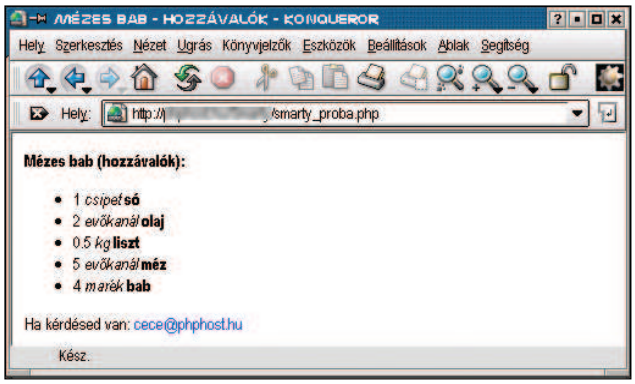
kerülésével a telepítési folyamatnak vége is, a Smarty harcra kész. Ismerkedjünk meg vele közelebbről!

Ismerkedés a lehetőségekkel

Mielőtt bármiféle használati útmutató megfogalmazásába kezdenék, nevezzünk meg egy példát, ami a Smarty leggyakrabban alkalmazott lehetőségeit mutatja be az 1. és 2. lista (megtalálható a 46. CD Magazin/PHP könyvtárában). Látható, hogy a PHP-kódban három sort minden esetben el kell helyeznünk:

```
require 'Smarty.class.php';
$smarty = new Smarty;
$smarty->display('smarty_proba.tpl');
```

Az első kettő gondoskodik a Smarty-motor életre keltéséről. Ezeket követi a sok `assign()` hívogatása, s legvégül a `display()` jeleníti meg az oldalt a megadott adatok és a sablonállomány alapján. A legtöbb esetben e három sor és a feladathoz szükséges számú `assign()` elég is, de talán senki sem csodálkozik, ha azt mondom, hogy rengeteg dolgot lehet még finomhangolni különféle függvényekkel és értékek beállításával. Ezekről azonban később szólok. Egyelőre az `assign()` használatára lesz csak szükség. Két adatot kell közölnünk minden ilyen érték-hozzárendelésénél: egyrészt meg kell adni azt a hivatkozási nevet, amellyel a sablonállományban a hozzárendelt értékre mutathatunk; második adatként pedig magát az adatot kell közölnünk. Erre több példát is mutat az 1. és 2.



És ami mindebből lesz

listán látható mintaprogram, például a

```
$smarty->assign('nev', 'MØzes bab');
```

sor a `$nev` Smarty változóhoz rendeli a „Mézes bab” értéket. Ez az adat természetesen jöhetett volna éppenséggel egy adatbázisból vagy bármilyen más adatforrásból is. Az is látható a példából, hogy nemcsak egyszerű adatok, de összetett tömbök is hozzárendelhetők a sablonhoz, és ezek tartalma teljes mértékben feldolgozható a sablonnyelv képességeivel. Az idén februárban kiadott 2.4.0-s változattól kezdve pedig már az objektumok adatainak kezelésével sincs gond.

Egyszerű behelyettesítések

A példából kitűnik, hogy a Smarty sabloncímkéket { } (kapcsos zárójelek) között kell elhelyezni. A legegyszerűbb és legtöbbet alkalmazott ezek közül a változóérték behelyettesítésére szolgáló címke. Ekkor a határolójelek közé a változó nevét kell elhelyeznünk, a dollárjellel megfejelve. Ezek a nevek azok, amelyeket az `assign()` függvények hívásakor az első értének adtunk. Ilyen alkalmazást mutat be a példában a `{$nev}` címke beillesztése.

Határozatlan hosszú listák

Szintén sűrűn alkalmazott, de egy fokkal bonyolultabb feladat a `{section}` címke alkalmazása. Erre akkor lehet szükség, ha listaszerű adatsort kell egységes arculat szerint megjelenítenünk anélkül, hogy a sablon tervezésekor tudnánk, valójában hány elemből is fog állni a lista. Jó példa erre a webes levelezőprogramok üzenetlistája. A sablonban az jelenik meg többszörözötten, ámde más-más adatokkal, ami a `{section}` és a `{/section}` címkék között található. Ezek a listák korlátlan mélységben egymásba is ágyazhatók. A `section` címke példánkban való alkalmazása azt mutatja be, hogyan is kell értékeket adni az egyes utasításokhoz. Ezeket { } jelek között, a `section`-től és egymástól is szóközökkel elválasztva lehet sorban megadni, a szokványos jelölőnyelvekhez hasonlóan. A `section` címkéhez rendelhető értékek a következők lehetnek:

- *name*: kötelezően megadandó adat, az adott `section` neve. Elnevezése a több egymásba ágyazható lista miatt szükséges. E névvel mutathatunk a végigzongorázott tömb pillanatnyi elemére (ahhoz hasonlóan, ahogy azt egy ciklusváltozóval tennénk).
- *loop*: szintén kötelező megadni, az itt megadott tömbből veszi az adatot arra nézve, hogy mekkora adathalmazon kell magát átrágnia.

- *start*: a listázás alapesetben a 0. tömbelemnél kezdődik, de ez itt módosítható. Lehetséges negatív egész érték megadása is, ekkor – a `substr()` működéséhez hasonlatosan – hátulról visszafelé számolva állapítja meg a kiindulópontot, azaz egy tízelemű tömb esetében a -3 érték a hetedik elemtől való kezdést jelenti.
- *step*: egy kis emlék a régi szép Basic-időkből. Ezáltal adható meg az a lépésköz, ahányasával majd végigugrálunk a tömbelemeken. Itt is meg lehet adni negatív értéket, ebben az esetben a megadott lépésnagyságban visszafelé fogunk lépkedni. Ha nem közlünk ilyen adatot, egyesével lépked előre.
- *max*: nem a legnagyobb tömbindexet, hanem azt szabályozza, hogy legfeljebb hány elem jelenjen meg a listában. Példaképpen: ha egy listát a `start=4 step=3 max=6` értékekkel indítunk el, az a 4., 7., 10., 13., 16., 19. elemeket fogja legfeljebb megjeleníteni.
- *show*: az adott lista látható legyen-e vagy sem. Egy `false` érték hozzárendelésével megtiltható a megjelenése.

Ilyen listázásoknál könnyedén előfordulhat olyan helyzet is, amikor valamilyen okból nincs mit listázni. Ilyenkor zavaró lehet, ha semmiféle visszajelzés nincs. Az efféle helyzetek kezelésére használható a `{sectionelse}` címke. Ha még a `{/section}` bezáró címke előtt letesszük, a közte és a záró-címke közt megadott szöveg jelenik meg abban az esetben, ha a ciklusban megadott tömb üres vagy egyéb okok (például a tömb utolsó eleme mögé mutató `start` érték) miatt nincs megjeleníthető adat (lásd a 3. listát a 46. CD Magazin/PHP könyvtárában).

Hivatkozás tömbelemekre

Beszéljünk egy kicsit a példában látható összetettebb változóérték behelyettesítéseiről, amik a lista előállításakor kerültek alkalmazásra. Mivel lehetőség nyílik tömböknek sablonhoz történő rendelésére, annak elemeit valahogy meg kell címezni. Ez egyszerű tömböknél a hagyományosnak mondható `$tomb[12]` formában adható meg, ami a `$tomb` nevű tömbváltozó 12-es indexén található adatra mutat. Többdimenziós tömbök esetén több szögletes zárójeles elem elhelyezése is elképzelhető.
A hagyományos PHP-formától eltérőbb módon zajlik az asszociatív tömbök kezelése. Ebben az esetben nem kapcsos zárójellel kell jelölnünk a megcélzott tömbelemet, hanem – egy ponttal elválasztva – annak nevével hivatkozunk. A példában szereplő `{$hozzavalok[i].mennyiseg}` tehát a `$hozzavalok` tömb i-edik tömbindexén ülő asszociatív tömb `mennyiseg` nevű elemére mutat.

Nem folytonos tömb listázása

A `{section}` egyik kellemetlen tulajdonsága az, hogy nem képes olyan tömböket végignyálazni, amelyekben „lyukak” vannak, azaz nem folytonos a tömbindex. Ez a helyzet azért áll fenn, mert ennek megvalósítására egy egyszerű `for` ciklus szolgál. Ha egy ilyen nehézséget okozó tömböt kell láthatóvá tennünk, jó szolgálatot fog tenni a `{foreach}`.
A `foreach` előnye nem csak a töredezett tömbök jó kezelésében rejlik – a `section`-nél is egyszerűbb a használata. Az egyszerűbb forma hátulütője viszont az, hogy menet közben nem áll rendelkezésünkre adat arról, hogy éppen melyik elemen állunk. Ehelyett az adott tömb pillanatnyi elemeit kapjuk meg. Ez akkor gond, ha az adatokat több tömbből kell összeszednünk. Természetesen a legtöbb ilyen nehézség okos programozással megelőzhető, így a `foreach` a legtöbb esetben jó választás.

A `{section}`-höz hasonlatosan itt is lehetőség nyílik az üres tömbök lekezelésére – a `{foreachelse}` hadrendbe állításával. A következő adatok közölhetők a `foreach` lista indításakor:

- *from*: annak a tömbnek a neve, amelyből az adatokat nyerjük. Mérete egyértelműen megadja a lista hosszát is, nincs lehetőség kezdőelem megadására vagy a legnagyobb elemszám korlátozására.
- *item*: ezen a változónéven érhető majd el a pillanatnyi tömbelem.
- *key*: a pillanatnyi tömbelem kulcsa, ha szükséges.
- *name*: erre akkor lehet szükség, ha az adott `{foreach}` tulajdonságaihoz hozzá akarunk férni. Ennek módjáról a következő cikkben lesz szó.

A Foreach alkalmazásával a hozzávalók listázása a 4. listán (46. CD Magazin/PHP könyvtár) látható módon festene.

A levélcímek védelme

Az utolsó elem a példában egy levélcímet megjelenítő parancs, a `{mailto}`. Hasonló látványt elérhettünk volna el a következő utasítás alkalmazásával is:

```
<a href="mailto:{$email}">{$email}</a>
```

Ennél a formánál a Smarty `mailto` címkéjének alkalmazása nemcsak szebb formájú, de használatával a kéretlen levelekhez címlistát gyűjtő robotok alkalmatlankodása is kivédhető.

A `{mailto}` címke megadható értékei közül a fontosabbak:
- *address*: a levélcím, amire a hivatkozás mutat.
- *text*: alapesetben a hivatkozás szövege a levélcím lesz, de itt átírható.
- *encode*: kéretlen levelek küldésére szakosodott kalózok adatgyűjtő robotjainak kijátszására szolgáló kódolási módozat kiválasztása. Alapesetben nem kódol, de „hex” és „javascript” kódolás is kérhető. Ezekkel a robotok már nem egykönnyen bánnak el.

Végszó

Első Smarty témájú cikkembe ennyi fért bele. A májusi számban tovább folytatjuk a felfedezést, további parancsokkal, szűrőkkel és adatmódosítókkal ismerkedünk meg. Sor kerül a Smarty bővítésének lehetőségeire és a gyorstárazás használatára is. Addig is sok szalonnasütést!

Heiglig (Cece) Szabolcs (cece@phphost.hu)
Veszprémben él. Hobbija, kedvenc időtöltése és munkája is a programozás. Szabadidejében hajlamos kerékpárra pattanni, vagy baráti társaságban szerepjátékokkal foglalkozni.

KAPCSOLÓDÓ CÍMEK

A Smarty hivatalos honlapja
➲ http://www.smarty.php.net
Sebességmániásoknak
➲ http://www.massassi.com/bTemplate/benchmarks/
➲ http://www.webmedic.net/webify_bench/tp1_bench_1_1.php

Unicode

A Unicode elengedhetetlen a nemzetközi webfejlesztés esetében, és új kérdéseket is felvet.

Mivel az Egyesült Államok északkeleti részén nőttem fel, az angolon kívül soha nem kellett más nyelven beszélnem. Angolul olvastam, angolul beszéltem, angolul írtam és angolul bonyolítottam le az üzleteket. Ugyanilyen cipőben jártak azok a mérnökök, akik 1968-ban létrehozták az ASCII kódot, s akik biztosak voltak benne, hogy 128 ASCII-karakter elégséges lesz az angol nyelvű dokumentumokhoz. Amíg a szabványos ASCII-karaktereknél maradtunk, nyugodtan másolhattunk fájlokat egyik gépről a másikra, és nem kellett attól tartanunk, hogy esetleg összezavarodnak.
Az ASCII nagyszerű volt a maga idejében, csakhogy azok az emberek, akik franciául, spanyolul és más nyugat-európai nyelveken beszéltek, hamar rájöttek, hogy ez így nem felel meg az ő igényeiknek. Akik ezeken a nyelveken használták számítógépüket, szerették volna helyesen viszontlátni mellékjeles betűiket. Így aztán idővel a 7-bites ASCII kód 8-bites kiterjesztett ASCII kóddá változott, s kibővült néhány, a nyugat-európai íráskép megjelenítéséhez szükséges különleges betűvel és jellel. Csakhogy mivel a kiterjesztett ASCII-t soha nem nyilvánították szabvánnyá, számtalan különféle, nem együttműködő ASCII-kódkiterjesztés látott napvilágot. A Windows is saját kiterjesztést készített, akárcsak a Macintosh és a NeXTSTEP operációs rendszer készítői. Így aztán hiába tudunk franciául dokumentumot írni Windows alatt, ha Macintosh alatt is használni szeretnénk, át kell alakítanunk, máskülönben a fogadó gépen a hibásan értelmezett bájtok miatt az egyébként tökéletes francia szövegünk inkább francia pirítósra (bundás kenyér) fog emlékeztetni.
Végül a nemzetközi szabványok győzedelmeskedtek, legalábbis bizonyos területeken, és megszületett a szabvány, amelyet formálisan ISO-8859-1 néven, kötetlenül pedig Latin-1-ként ismerünk. A számítógépgyártók ezek után már képesek voltak nyugat-európai dokumentumokat cserélni anélkül, hogy a karakterek összekeveredésétől kellett volna tartaniuk. Természetesen ezt úgy érték el, hogy a karakterbájtoknak mind a 8 bitjét kihasználták, 128-ról 256-ra emelve a lehetséges karakterek számát.
Csakhogy ez sem oldott meg minden gondot, például a héber nyelvűeknek saját szabványuk volt ISO-8859-8 néven, amely a 0–127 kódig a Latin-1-gyel azonos, viszont erősen eltér a 128–256 helyeken. A héberül írt dokumentumok Latin-1 alapú számítógépen megjelenítve úgy néznek ki, mint valami betűhelyettesítős rejtvény, ami rossz ábécéből vette a betűket.
Ez gyakorlatilag azt jelenti, hogy az ISO-8859 szabványsorozattal nem tudunk olyan dokumentumot írni, ami egyszerre tartalmaz angol, héber és francia karaktereket. Tulajdonképpen ez érthető is, hiszen egyetlen bájton mindössze 256-féle karakterrel tudunk játszani. Komoly nehézséggel kell azonban szembenézniük azoknak, akik több mint két nyelven dolgoznak. A dolgok akkor kezdenek igazán cifrává válni, amikor a lapot egyszerre szeretnénk angolul, héberül, franciául és kínaiul megjeleníteni. Ugyanis a kínai nyelvben ideogrammák tízezrei szerepelnek, és akkor még nem is beszéltünk a japán és egyéb nyelvekről!

Máris megérkeztünk a Unicode-hoz, a következő század ASCII-táblájához. Az ASCII-höz hasonlóan a Unicode is számokat rendel a betűkhöz, számjegyekhez és szimbólumokhoz. Az ASCII-val ellentétben azonban a Unicode elég helyet hagy az emberi nem által valaha kitalált összes szimbólum számára. Ez egyben azt is jelenti, hogy a Unicode-dokumentum akárhány nyelv akárhány karakterét tartalmazhatja, és eközben az ütközések miatt sem kell aggódnunk. A Unicode egyúttal kezelni tud néhány olyan nehézséget is, amiről az ASCII soha nem is álmodhatott. Például vegyítheti a karaktereket (ékezetekhez és egyéb mellékjelekhez), és kezelheti az irányultsági kérdéseket (azoknál a nyelveknél, amelyek nem balról jobbra olvasandók).
A Unicode már közel egy évtizede létezik, de a webalkalmazások világában csak most kezd népszerűvé és támogatottá válni. E hónapban azt vizsgáljuk meg, hogy milyen hatással van a Unicode a webfejlesztésre. Mit kell figyelembe vennünk? Mi miatt kell aggódnunk? És végül hogyan lehetünk úrrá az Unicode-dal kapcsolatos nehézségeken?

Bevezetés a Unicode világába

A Unicode, akárcsak az ASCII, egyedi számokat rendel minden betűhöz, számhoz, szimbólumhoz és egyéb vezérlőkarakterhez. De ahogy azt már korábban említettük, a Unicode a valaha alkotott összes szimbólumra és karakterkészletre kiterjed. Így aztán Unicode-ot használva olyan dokumentumokat is készíthetünk, amelyek egyaránt használnak angol, orosz, japán és arab karaktereket, s ahol minden egyes karakter egyértelműen megkülönböztethető a másiktól.
De hogyan változtathatjuk át ezeket az egyedi számokat (amelyeket a Unicode világban kódpontoknak neveznek) bitekké és bájtokká? Az ASCII kódolása eléggé magától értetődő; mindössze 127 karakter létezik (vagy 256, ha a különféle kiterjesztéseket is alkalmazzuk), így minden egyes ASCII-karakter elfér egyetlen bájtban. C-programozók bizonyára tudják, hogy a `char` (karakter) adattípus egy 8 bit hosszú egész.
A legnyilvánvalóbb megoldás, ha állandó hosszúságú, több bájtos kódolást használunk Unicode-karaktereinkhez. A UCS-2 éppen ilyen kódolás, amely két bájtot használ, hogy leírja vele a 65 536 legalapvetőbb Unicode-karaktert. (Van néhány kiterjesztett karakter, amely további bájtokat igényel, de ebbe most nem megyünk bele.) A UCS-2 minden karakterhez egyetlen kétbájtos kódot rendel. A dokumentumok így a felhasznált nyelvtől függetlenül egyforma hosszúak lesznek, a programok pedig könnyen kiszámíthatják a szükséges bájtok számát – egyszerűen megkétszerezik a karakterek darabszámát.
A Microsoft korszerű operációs rendszerei UCS-2-t használnak, ahogy azt nyilván megfigyelhettük már, ha ennek a rendszernek a felhasználóival cseréltünk dokumentumokat.
A UCS-2 esetében azonban alapvető gondunk támadhat, nevezetesen az, hogy nem működik együtt az ASCII kódolással. Ha százezer ASCII kódolású dokumentumunk van, vala-

mennyit át kell fordítanunk UCS-2-vé, ha helyesen akarjuk őket olvasni. Figyelembe véve, hogy a legtöbb korszerű program ASCII-val dolgozik, az ilyesfajta visszafelé történő együttműködés hiánya elég nagy gond.
Megjelent tehát a UTF-8, a változó hosszúságú Unicode kódolás. Ahogy ugyanazokat a számokat a római és az arab számok másként jelölik, a UTF-8 és a UCS-2 is ugyanannak a háttérben maradó Unicode-karakterkészletnek a különféle kódolása. Csak míg minden egyes UCS-2-karakter két bájtot használ, a UTF-8-karakterek egytől négy bájtig bármennyit használhatnak.
Az egybájtos UTF-8-karakterek azonosak az ASCII kóddal, ami azt jelenti, hogy egy helyes ASCII-dokumentum egyben helyes UTF-8-dokumentum is. Csakhogy a Latin-1 és egyéb 8-bites karakterkészletek már nem működnek együtt a UTF-8-cal; a már létező Latin-1-dokumentumokat nemcsak, hogy át kell alakítanunk, de a méretük akár meg is kétszereződhet.
Unix-és Linux-rendszereken és az általam általában használt szabványokban és nyílt forrású programokban a UTF-8 az előnyben részesített kódolás. A Perl, a Python, a Tcl és a Java mind UTF-8 kódolással rögzítik a szövegeket. A PostgreSQL évek óta támogatja a UTF-8-at, és nemrégiben bekerült a Unicode-támogatás a MySQL 4.1-be is, amelyet a következő hónapban fognak alfakódként kihozni.
Létező rendszerekbe beépíteni a Unicode-támogatást héroszi erőfeszítést jelent, amiért a fejlesztőket hatalmas dicséret illeti meg. A fejlesztőknek nemcsak a több bájtos karakterek kezelését kell bevinniük, de az adatbázisoknak és a nyelveknek a szabványos kifejezéseket és rendezési operátorokat is támogatniuk kell, ezeket pedig nem éppen egyszerű elkészíteni.

A Unicode és a HTTP

Végigvettük az alapokat, most nézzük meg, hogyan továbbítódik egy Unicode-dokumentum a weben. Az alapvető kérdés az, hogy amikor böngészőnk megkapja a dokumentumot, honnan fogja tudni, hogy a bájtokat Latin-1-es, Big-5 Chinese vagy UTF-8 kódolással kódolták-e?
A válasz a Content-type HTTP-fejlécben rejtezik. Valahányszor a HTTP-kiszolgáló elküld egy dokumentumot a böngészőnek, az elküldött tartalmat bizonyos MIME stílusú meghatározással azonosítja, például `text/html`, `image/png` vagy `application/msword`. Ha egy JPEG-képet kapunk (image/jpeg), azt csak egyféleképpen jeleníthetjük meg. Ezzel szemben ha egy HTML-dokumentumot kapunk (text/html), a Content-type fejlécben meg kell adni a karakterkészletet, illetve a felhasznált kódolás típusát. Ezt a fejléc végéhez csatolt `charset=` meghatározás felvételével tehetjük meg, ami a típust elválasztja a karakterkészlettől. Például:

```
Content-type: text/html; charset=utf-8
```

A figyelmesen olvasók jogosan mondhatják, hogy a UTF-8 nem karakterkészlet (charset), hanem kódolás (encoding). Sajnos már késő, ez ellen nem tehetünk semmit. Ez is olyasmi, mint a HTTP-szabvány elgépelt `referer` kulcsszava, amelyet helyesen `referrer`-nek kellene írni; mindenki tudja hogy rossz, de félnek elrontani a már létező programokat.
Ha nem adunk meg Content-type-ot, akkor Latin-1-et kell feltételezni. Továbbá, ha a Content-type nincs megadva, az egyes dokumentumok ezt az értéket metatagjaikban megadhatják (vagy felülbírálhatják). A metatagok ellenben nem bírálhatják felül a közvetlenül megadott kódolást.
Ha elkezdünk dolgozni a különféle kódolásokkal, hamarosan találkozni fogunk olyan HTTP-kiszolgálóval, amelyik rosszul lett beállítva, és hibás karakterkészletet továbbít a Content-type fejlécben. Ezt könnyen ellenőrizhetjük a Perl LWP (Library for Web Programming, azaz webprogramozói könyvtár) felhasználásával, amely számos, a webfejlesztésben jól használható parancssoros programot tartalmaz. Írjuk be:

```
$ HEAD http://yad2yad.huji.ac.il/
```

Az én linuxos gépemen a fentiek begépelése után a megnevezett lapról a következő HTTP-válaszfejléc érkezett:

```
200 OK
Cache-Control: max-age=0
Connection: close
Date: Tue, 10 Dec 2002 08:38:37 GMT
Server: AOLserver/3.3.1+ad13
Content-Type: text/html; charset=utf-8
```

Mint láthatjuk, a Content-type fejléc állítása szerint a dokumentum UTF-8 kódolású.
A Mozilla és más korszerű böngészők lehetővé teszik, hogy a felhasználó felülbírálhassa az alapértelmezett kódolást. Bár általában az egyszerű felhasználóknak nemigen van szükségük ilyesmire, webfejlesztés közben gyakran igen hasznosnak találtam ezt a szolgáltatást.

A Unicode és a HTML

Jó tudni, hogy UTF-8-dokumentumokat is átküldhetünk a HTTP-n, de nem ártana, ha lenne ehhez néhány UTF-8-dokumentumunk is. Figyelembe véve, hogy az ASCII-dokumentumok egyben UTF-8-dokumentumok is, érvényes UTF-8-dokumentumot létrehozni nem nagy ördöngösség, egyszerűen kizárólag ASCII-karaktereket kell használnunk. De mi a helyzet, ha olyan HTML-lapokat szeretnénk létrehozni, amik héber vagy görög betűket tartalmaznak? A dolgok ekkor kezdenek érdekessé és bonyolulttá válni.
Alapvetően kétféleképpen tudunk Unicode-karaktereket vinni HTML-dokumentumunkba: az első lehetőség, hogy valamilyen UTF-8 támogatású szövegszerkesztővel gépeljük be a karaktereket. Például a GNU Emacs különféle billentyűzetopciókkal teszi lehetővé, hogy a szöveget bevigyük, majd a dokumentumot tetszőleges, többek között UTF-8 kódolással menthessük. Ha kínai szöveget akarok Latin-1 kódolással menteni, az Emacs tiltakozni fog, és figyelmeztet, hogy a dokumentum olyan karaktereket tartalmaz, amelyek a Latin-1 kódolás alatt nem léteznek. Sajnos azok számára, akik hozzám hasonlóan héberül szeretnének írni, az Emacs sem tökéletes megoldás, mivel egyelőre nem támogatja a jobbról balra történő írást.
Jobb választás, és idővel egyre hatásosabbá váló megoldás a Yudit nevű UTF-8-támogatású nyílt forrású szövegszerkesztő, amely számos különböző nyelvet és irányt kezel. Beletelhet egy kis időbe, amíg kiismerjük a Yuditot, de megéri a fáradtságot. A Yudit, akárcsak az Emacs, tetszés szerint bármilyen karakter beütését engedélyezi, még akkor is, ha az operációs rendszerünk vagy a billentyűzetünk az összes használni kívánt nyelvet közvetlenül nem is támogatja.
Az Emacs és a Yudit egyaránt jó választás, ha Linux alatt dolgozunk, hajlandóak vagyunk bütykölgetni, és nem bánjuk, ha kézzel kell megírnunk a HTML-t. Sajnos szinte az összes általam ismert grafikus tervező más rendszereken dolgozik, és igencsak nehéz rávenni őket, hogy UTF-8 kezelésére is képes HTML-szerkesztőket használjanak.
Szerencsére a Mozilla nemcsak webböngészőt, hanem egy

teljes értékű HTML-szerkesztőt is tartalmaz. Mint az várható is, a Mozilla szerkesztőmodulja néha kicsit nehézkes, de a legtöbb feladatot képes jól kezelni.
A másik lehetőség, hogy HTML-elemeket használunk. A legismertebb ilyen elemek a < a &gt és az & amelyek a <, > és & jelek beszúrását teszik lehetővé HTML-dokumentumainkban, így nem kell attól tartanunk, hogy esetleg tagokként értelmeződnek.
A korszerű böngészők nemcsak az olyan elemeket értik, mint a © (a copyrightjel), hanem a Unicode-karakterek teljes kínálatát is. Dokumentumainkban bármilyen Unicode-karakterre hivatkozhatunk a &#XXXX; alakban, ahol az XXXX helyére a karakter decimális kódját kell írnunk. Például a következő HTML-dokumentum az én nevemet jeleníti meg héber nyelven, Unicode-elemek segítségével:

```
<html>
     <head><title>Reuven neve</title></head>

<body><p>&#1512;&#1488;&#1493;&#1489;&#1503;</p>
     </body>
</html>
```

A fenti dokumentum elkészítéséhez nem volt szükségünk Unicode-alkalmas szövegszerkesztőre, és minden korszerű böngészőn jól jelenik meg, a HTTP válaszfejlécében beállított Content-type értéktől függetlenül. Ugyanakkor az ilyen elemeket tartalmazó fájlok szerkesztése jobb esetben is fárasztó és bonyolult. Sajnos a Microsoft Word nemzetközi változatainak mentés HTML-ként képessége kiterjedten használja ezt a lehetőséget, ami a Word-felhasználóknak ugyan egyszerűvé teszi a Unicode-alkalmas HTML-dokumentumok mentését de jelentősen megnehezíti azok életét, akik utólag szeretnék szerkeszteni őket.

Csapdák

Mint azt már korábban jeleztem, a Unicode összetett szabvány, és jó időbe telt, mire a különféle nyelvek és módszerek támogatni tudták. Például a Perl 5.6.x belső Unicode-támogatással bír, de a ki- és bemeneti műveletek ezt nem tudják könnyen kihasználni, így a támogatás tulajdonképpen hasznavehetetlen. A Perl 5.8 ezzel szemben már kitűnő Unicode-támogatással rendelkezik, amelyben a fejlesztők már a Unicode-tulajdonságoktól függő szabványos kifejezéseket is írhatnak.
Azért maradtak még nehézségek: az egyik komoly gond, amivel a fejlesztőknek szembe kell nézniük, a bemenetkódolás és tárolt kódolás ellentétes igényeivel kapcsolatos. Ilyen esetre példa, amikor a terminálunk Latin-1 kódolást használ, ugyanakkor háttérrendszer UTF-8 rendszerben működik. Az ilyesfajta elrendezés azt eredményezi, hogy továbbra is használhatjuk régi (nem Unicode) terminálprogramunkat és betűtípusainkat, ugyanakkor csatlakozni tudunk és használhatjuk a Unicode-alkalmas háttérrendszert.
Néhány más megvalósításnak is akadnak hiányosságai, amelyek esetleg nem is tűnnek nyilvánvalónak, amikor elkezdünk dolgozni a projekttel. Például nemrégiben egy J2EE-projekten dolgoztam, amely PostgreSQL-t használt háttérként és minden karaktert Unicode-ban tárolt. Minden kitűnően működött egészen addig, amíg el nem határoztuk, hogy a felhasználó által beírt szöveget kis- és nagybetűk megkülönböztetése nélkül hasonlítjuk össze az adatbázisban tárolt szöveggel. Sajnos az általunk használt PostgreSQL-függvény a Unicode-karakterek esetében nem kezelte helyesen a kis- és nagybetűk azonosságát (case insensitivity). Végül ugyan meg tudtuk oldani a feladatot, de elég kínos és kiábrándító volt ilyesmivel szembetalálkozni.
Az összehasonlítás vagy a rendezés szintén kemény dió – olyasmi, ami több alkalommal is megizzasztott. A Unicode ugyan meghatároz karakterkészletet, de nem jelöli meg, hogyan kell ebben a készletben a karaktereket sorba állítani. Például egészen mostanáig a „ch” külön helyet foglalt el a spanyolul beszélő országok ábécéjében; ugyanakkor ez az angol, német és francia anyanyelvű országokra nem volt igaz. A rendezési módszer tehát nemcsak a karakterkészlettől, hanem a karakterkészletet alkalmazó helyi beállításoktól is függ. Lehet, hogy egy kicsit kísérleteznünk kell a `LANG` és az `LC_ALL` környezetei változókkal (többek között), hogy végül az elvárásainknak megfelelően működjenek a dolgok.

Összefoglalás

A jövő egyértelműen a Unicode-é; bizonyos szinten a legtöbb operációs rendszer már most is támogatja, és egyre erősebb szabvánnyá válik a számítástechnikában. Sajnos a Unicode eljövetelével elfelejthetjük a karakterek és a bájtok egyenlőségének régi programozói gyakorlatát, és jó néhány új, összetett feladat is felbukkan.
Ha csak egyetlen nyelvet akarunk használni a weblapunkon, szerencsésnek tarthatjuk magunkat. De ha csak egyetlen nem ASCII-karaktert is fel akarunk használni, hamar mélyen a Unicode világában találhatjuk magunkat. Jobb mihamarabb megismerni ezt a technológiát, hiszen ha lassan is, de bizonyosan elkerül valamennyi nyílt forráskódú rendszerbe és szabványba.

Linux Journal 2003. március, 107. szám

Reuven M. Lerner (➲ http:/www.lerner.co.il/atf) nyílt forrású programokra, valamint web- és adatbázis-alkalmazásokra szakosodott tanácsadó. Könyve, a Core Perl, 2002 januárjában jelent meg a Prentice Hall gondozásában. Reuven feleségével és lányával Izraelben, Modi'in-ben él.

KAPCSOLÓDÓ CÍMEK

Karakterkészletekről és kódolásokról szóló kitűnő ismertetőt találunk *Andy Deitsch* és *David Czarnecki* Java Internationalization című művének első fejezetében, ami az O'Reilly kiadó gondozásában jelent meg. Eleve érdekes dolog olvasni arról, milyen sokféle írásmódot használnak az emberek, egyben megtanuljuk értékelni a Unicode konzorcium erőfeszítéseit is, amelyeket e karakterkészlet létrehozása során kellett megtennie.
A Yudit szövegszerkesztő és sok egyéb Unicode témájú érdekes hivatkozás található a ➲ http://www.yudit.org címen.
A ➲ http://www.gnu.org/directory/recode.html címen találjuk a GNU Recode programot, ami lehetővé teszi, hogy dokumentumokat alakítsunk át különféle karakterkészletek és kódolások közt. Mikor Unicode vonatkozású oldalakkal dolgozom, a Recode nem hiányozhat az eszköztáramból.

A Mathematica 4.2

Nem könnyű körülírni, hogy mi is a Mathematica, mire használható leginkább, hiszen számos különböző célra alkalmas.

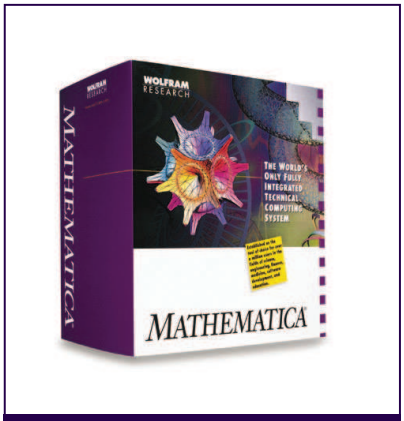
Aa egyszer valaki, aki még soha sem hallott róla, megkérdezné: mégis, mi ez a Mathematica, azt válaszolnám: „a műszaki számítások svájci bicskája". Lehet, hogy ezzel a kérdés még nyitva maradna, de aki egyszer megismeri a programot, tudni fogja, hogyan kell érteni ezt a kijelentést.

Hogy egy kicsit kifejtsem a svájcibicska-hasonlatot, megemlítek néhány olyan területet, ahol a Mathematicát sikerrel alkalmazzák napjainkban: mérnöki, fizikai számítások, számítástudomány, könyvkiadás, pénzügy, illetve közgazdaságtan, matematika, társadalomtudományok.

A Mathematica – alkalmazási lehetőségeihez hasonlóan – a támogatott eszközök és operációs rendszerek tekintetében is sokoldalúságot mutat; futtatható Linux, Mac OS X, Mac OS 8.1 vagy újabb, Windows, Solaris, Tru64, HP-UX, AIX és Irix operációs rendszeren egyaránt.

Ha a Mathematica csak Linuxra írt változatára összpontosítunk, látni fogjuk, hogy a Wolfram Research számára a Linux nem csupán az egyik operációs rendszer az összes közül, amelyikre a programot egyszerűen át kell ültetni. A Mathematica magját Linux alatt fejlesztették, és csak ezután ültették át a többi rendszerre, így nem csoda, hogy a Mathematica Linux-változata igazából négy különböző számítógép-kiépítésen működik: IA-32 (x86), PowerPC, Alpha és IA-64 (Itanium). Sőt Linux-telepekhez is létezik egy külön változata, a gridMathematica.

Kétségtelen, hogy a Wolfram Research számára a mind a profik, mind pedig a diákok piacot jelentenek. A kereskedelmi termék megterhelő lehet az otthoni felhasználók pénztárcájának, de vállalati vagy kutatói környezetben alkalmazva bőven visszahozza az árát. Másrészt a Mathematica diákoknak szánt változata igazán előnyös feltételekkel férhető hozzá, és pontosan ugyanazokkal a képességekkel rendelkezik, mint a kereskedelmi változat.



Termékjellemzők

Gyártó: Wolfram Research
Honlap: ➲ http://www.wolfram.com
Kiskereskedelmi ár: 1880 dollár
Oktatói, kutatói ár: 895 dollár
Ára diákok számára: 139 dollár

A csomag összetevői

A Mathematica 4.2 csomagban az alábbi összetevőket találjuk: „Installing Mathematica 4.2 for Unix and Linux" (A Mathematica telepítése Lunixra és Unixra), „Getting Started with Mathematica" (Kezdeti lépések a Mathematicával), The Mathematica Book (A Mathematica könyv) negyedik kiadása, „Standard Add-on Packages" (Általános kiegészítő csomagok) és a Mathematica 4.2 (a telepítő CD).

Az „Installing Mathematica 4.2 for Unix and Linux" című tájékoztató a Linux- és Unix-rendszerekre történő telepítést, valamint a korábbi változatokról való frissítést írja le.

A ***Kezdeti lépések a Mathematicával*** című kiadvány egy ötvenoldalas füzetke, ami a Mathematica alapjait ismerteti. Jó kiindulási pontot jelent a programmal most ismerkedők számára, gyorstalpaló tanolyam a Mathematica használatához. A tájékoztató számos olyan, a Világhálón elérhető anyagot is felsorol, amelyek az új felhasználók számára szinte nélkülözhetetlenek.

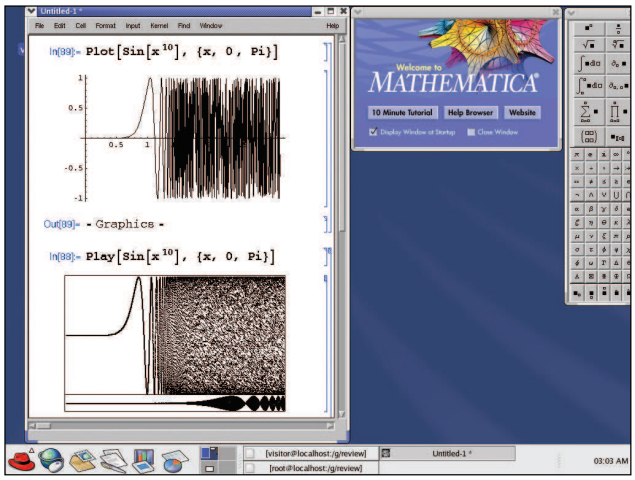
A Mathematica könyv körülbelül 1500 oldalon keresztül mutatja be a Mathematica függvényeit és képességeit. Ez a tárgyban felelhető legrészletesebb útmutató több adatot tartalmaz, semhogy egy ilyen rövidke ismertetőben áttekinthető lenne.

Az ***Általános kiegészítő csomagok*** pedig egy olyan ötszáz oldalas könyv, amelyik a kiegészítő csomagokban található ezernél több további függvényt ismerteti. Ezek a kiegészítő csomagok részei a Mathematica teljes változatának, és többek közt az alábbi témákkal foglalkoznak: grafika, geometria, hangtan, statisztika, mértékegységek, algebra, analízis, diszkrét matematika, numerikus módszertan. A könyv tárgya némileg ***A Mathematica könyv*** kiegészítésének tekinthető, úgy éreztem, hogy ténylegesen is a részét kellene képeznie, különösen, ha figyelembe vesszük, hogy a kiegészítő csomagok a Mathematica minden kiadásában megtalálhatók.

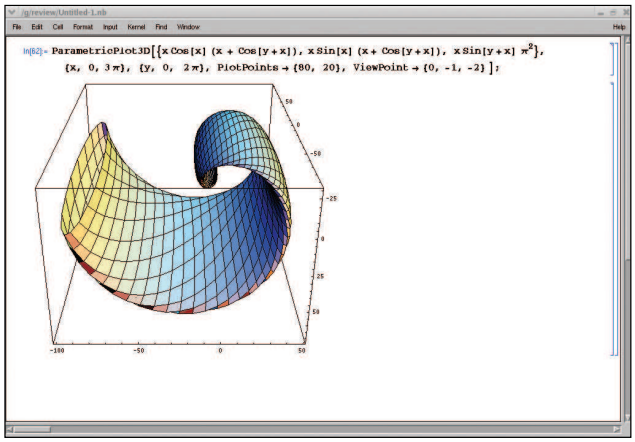
A Mathematica 4.2 Linuxra írt változata három különböző kiépítést támogat: az x86-t, az Alphát és a PowerPC-t (az Itaniumra írt változat csak nemrég látott napvilágot, így nem szerepelt a telepítő CD-n). A lemez tartalmazza a Mathematica könyvet és az Általános kiegészítő csomagok című könyveket is a Mathematica „Notebook" formátumában, ami böngészhető és kifejezések keresésére is lehetőséget nyújt. Ez a formátum jó referenciát jelent, amikor a Mathematicában dolgozunk, bármiről legyen is szó.

A program telepítése nem jelenthet gondot. Behelyezzük a CD-t az olvasónkba, az ***Installers*** könyvtárban megkeressük a megfelelő kiépítés alkönyvtárát, és futtatjuk a `MathInstaller` parancsfájlt. A telepítés végén meg kell adnunk a felhasználási szerződésünk számát és a hozzá tartozó jelszót. Akkor sincs baj, ha a telepítéskor még nem rendelkezünk ezekkel az adatokkal, ugyanis a program a futtatás során ismét kérni fogja.

Felhasználási szerződésszámunk mindig ugyanaz marad, még a program frissítésekor sem változik. Jelszavunkat a Wolfram Research állítja elő, amikor regisztráljuk a példányunkat. Ehhez a felhasználásiszerződés-számunkra és a Mathematica által programinduláskor előállított MathID-re van szükség. Saját tapasztalatomból kiindulva nem ajánlom a program



1. kép Néhány kétdimenziós grafika



2. kép A Mathematica 3D-képességeinek felfedezése

telepítését olyan gépre, amelyben gyakran változnak az alkatrészek. Ha ezek módosulnak, az nagy valószínűséggel a MathID változását is maga után vonja. Ugyanez igaz új rendszermagra történő frissítéskor. Ez azt jelenti, hogy fel kell venni a kapcsolatot a Wolfram Researchcsel és új jelszót kell kérnünk. Biztos vagyok benne, hogy ez a módszer megkíméli a céget néhány jogosulatlan felhasználótól, de meg kell mondanom, hogy elég kényelmetlennek találtam.

A Mathematicát a hátterében rejlő elv teszi igazán sokoldalú és hatékony számítási felületté. A programban mindent egy szimbolikus kifejezés képvisel, aminek a `Sin[x]` kifejezés egy jellemző példája. Ez a kulcsa a működésnek, mivel minden függvény értéket vehet és kaphat bármely más függvénytől. Ez annyi lehetőséget nyit meg, hogy valósággal már a puszta száma is elárasztott azoknak a dolgoknak, amiket a vizsgálatom során kipróbálhattam.

Hogy jobban el tudjuk képzelni a Mathematica „bármit bármihez" elvét, nézzük az *1. képen* lévő szinusz függvényt. A `Sin` függvény az `x10` változótól kapja a bemenetét és az eredményt a `Plot` függvénynek adja tovább. Ez a `Sin` által előállított értékek felrajzolását eredményezi. Ha úgy döntünk, hogy a rajz helyett valahogy másként szeretnénk az eredményt felhasználni, a kimenetet egyszerűen átadhatjuk egy másik függvénynek. Például a `Play` függvény a pillanatnyi hangot állítja elő, ami a képre kattintva hallható.

Mindenképpen meg akartam vizsgálni a Mathematica 3D-s képességeit is. A *2. képen* látható, hogy a szinusz és koszinusz függvényeket használom a `ParametricPlot3D` függvény bemeneteként. Ha befejeztem a kifejezés beírását és megnyomom a SHIFT-ENTER billentyűkombinációt, a program megrajzolja a függvény háromdimenziós képét. Ez csak néhány lehetőség a Mathematica 1100 beépített és 1000 kiegészítő függvénye közül, amelyek a 4.2 csomagban megtalálhatók. A vizsgálathoz rendelkezésemre álló idő rövidsége miatt úgy éreztem, hogy valójában a programban rejlő számítási lehetőségeket még nem is érintettem.

Nem csak arról van szó, hogy ***dr. Michael Trott***, aki Wolfram Research munkatársa és a Graphica (➲ http://www.gaphica.com) könyv szerzője, nagylelkűen rendelkezésemre bocsátotta a megjelenés előtt álló, „The Mathematica Guidebook Series" (Mathematica útmutatósorozat) teljes kéziratát (2 GB). Igazi élvezet volt azoknak a kifejezéseknek a böngészése és kipróbálása, amelyek azokat a nagyszerű képeket eredményezik, amik a Wolfram Research kiadványaiban is megjelennek – ilyen például A Mathematica könyv borítóján látható dodekaéder. Alig várom, hogy elkészüljön a könyvsorozat.

A Mathematica kipróbálása két különböző számítógép-kiépítésen, két operációs rendszer alatt folyt. Én egy x86-os gépen futó Linux alatt próbáltam ki a rendszert, míg ***Helen Moore*** matematikus egy Mac OS X rendszeren dolgozott. Bár Helen segített az eredmények értékelésében, ez az írás kizárólag az én véleményemet tartalmazza.

A két rendszeren a Mathematica lényegében ugyanúgy működött, Helennel együtt meg voltunk vele elégedve. Egyedül a regisztrációs és jelszó-előállító folyamatra haragudtunk egy kicsit. Ettől eltekintve bárkinek nyugodt szívvel ajánlhatjuk, dolgozzon akár Linuxon, akár Mac OS X alatt.

A Wolfram Research munkatársai készséggel segítettek bármilyen általam felvetett gond megoldásában (jó a kiszolgálás). A Mathematica for Linux nagyszerű program, talán a legjobb program, amit valaha Linuxon futtattam. A kipróbálás alatt magamról megfeledkezve órákat töltöttem a program képességeinek próbálgatásával, a lehetőségek felfedezésével. Helen szavait idézve: „A Mathematica remek program. Valószínűleg a legjobb kereskedelmi csomag szimbolikus és numerikus számítások végzésére."

Előnyök

- Széles körű alkalmazási lehetőségek
- Kiegészíthetőség
- A támogatott kiépítések nagy száma
- Részletes leírások

Hátrányok

- Regisztráció és jelszó-előállítás

Köszönetnyilvánítás

Köszönetemet fejezem ki ***dr. Helen Moore***-nak a kiértékelésben nyújtott segítségéért, ***dr. Michael Trott***-nak, hogy használhattam a jegyzeteit, ***Ben Wilson***-nak és ***P. J. Hinton***-nak, a Wolfram Research munkatársainak a segítségért és gyors válaszokért, amiket felvetett kérdéseimre adtak.

Linux Journal 2003. március, 107. szám

Dragan Stancevic

Huszas évei végén járó rendszermag- és hardvermérnök. Bár hivatását tekintve programmérnök, mély érdeklődést érez az alkalmazott fizika iránt, és az is köztudott róla, hogyha szabadidejében játszani kezd, akkor szinte megszállottként viselkedik.

Matematikai programok Linux alatt

Nincs olyan ember, akinek ne lenne valamilyen szintű kapcsolata a matematikával. Iskolai tanulmányaink alatt több év alatt jutunk el egyszerűbb és bonyolultabb feladatokon keresztül az elvonatkoztatás egyre magasabb fokáig.

Hétköznapokon az alapműveleteken kívül csak kevés matematikai ismeret szükséges, de a természettudományok, így a számítástechnika művelői számára is nélkülözhetetlen a magas szintű matematikai tudás.

Tanulóként vagy olyan szülőként, akinek a gyerekei matematikát tanulnak, bizony, nagy segítség lehet, ha az otthoni számítógépen matematikai programok is futnak. Pénzügyi kimutatásaink elkészítéséhez nagyszerű segítséget nyújthatnak a táblázatkezelő programok, amelyek szintén matematikai alapúak. Ha tudományos munkában veszünk részt, a matematikai tárgyú részfeladatok megoldásában is jól jöhet a számítógép, bizonyos szakterületeken a használata elengedhetetlen.

A Linux elsőrangú operációs rendszer matematikai programok futtatásához és fejlesztéséhez. Számos kereskedelmi célú matematikai programnak létezik linuxos változata, sőt sok program esetén éppen a Linux a fő fejlesztői operációs rendszer. Általános célú és egyedi, kifejezetten az adott szakterületre írt programok egyaránt rendelkezésünkre állnak. Itt most csak olyan programokról lesz szó, amelyek a szélesebb közönség számára is érdekesek lehetnek.

Szinte minden Linux-terjesztés tartalmaz néhány matematikai programot. Ezek közül bizonyosak az operációs rendszer működését a parancsértelmező szintjén segítik elő, ilyen például az `expr` és a `bc` program. (Ha valamit gyorsan kell kiszámolni, én nem is használok mást, csak `bc`-t. Az újabb Linuxokon a `man bc` parancssorral magyar nyelvű kézikönyvoldalhoz jutunk.) Más programokat jellemzően az ablakkezelő ***Oktatóprogramok/Matematika*** menüjéből indíthatunk, amennyiben az általunk használt Linux-változat ezt támogatja (a SuSE Linux például ilyen). Ha olyan programot szeretnénk használni, ami nincs meg terjesztésünkben, akkor azt az Internetről tudjuk a legkönnyebben beszerezni. Általános szabály, hogy a legfrissebb változat rendszerint csak forráskódban érhető el, többnyire ***.tar.gz*** formátumban. (Az ilyen fájlok telepítési módjáról a Linuxvilágban többször is olvashattunk már.) Néhány program további programokra vagy fejlesztői csomagokra (`*-devel`) épül; amennyiben hibajelzést kapnánk, próbálkozzunk Linux-rendszerünk frissítésével. (A rendszerfrissítés külön szakma, amivel még gyakorlott linuxosként is meggyűlhet a bajunk.)

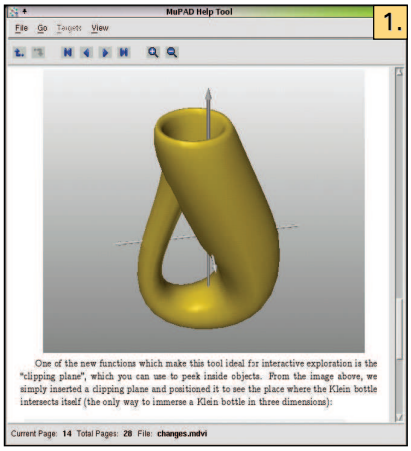
Az Internetet böngészve igen-igen sok matematikai programra bukkanhatunk, amelyeknek a minősége nagyon különböző lehet. Bizonyos programokat évek óta nem fejlesztenek már, másoknak naponta érkezik új változata. A SourceForge.net ***Software Map*** hivatkozására kattintva a ***Science/Mathematics*** kategóriában tevékenységük mértéke szerint látjuk a matematikai projekteket, ami jó megközelítést arra vonatkozóan, hogy milyen programokat érdemes időről időre kipróbálnunk. A

➲ http://www.gnu.org/directory/science/math cím kifejezetten a GNU projektbe tartozó fejlesztések listáját adja meg (ez lényegesen rövidebb lista). Hasznos hivatkozásgyűjtemény a

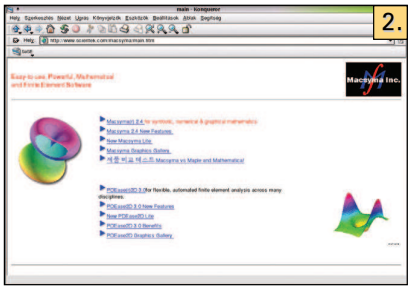
➲ http://richtech.ca/cgi-bin/seul/seulview.pl?category=Math cím is, ezt főként oktatási szempontok szerint állították össze a weblap karbantartói.

Komputeralgebrai rendszerek

Ma Magyarországon talán a Maple (➲ http://www.maplesoft.com) a legismertebb komputeralgebrai rendszer (Computer Algebra System, azaz CAS). A Maple és a hozzá hasonló programok óriási matematikai bázissal, párbeszédes módon segítik a feladatmegoldást. Fő erejük a pontos (nem kettedestört alapú kerekített) számolás, a számos beépített matematikai szolgáltatás és a könnyű kezelhetőség. (Ez utóbbihoz annyit hozzá kell tenni, hogy a programmal történő párbeszéd angol nyelvű, ami sok felhasználónak gondot okozhat.)



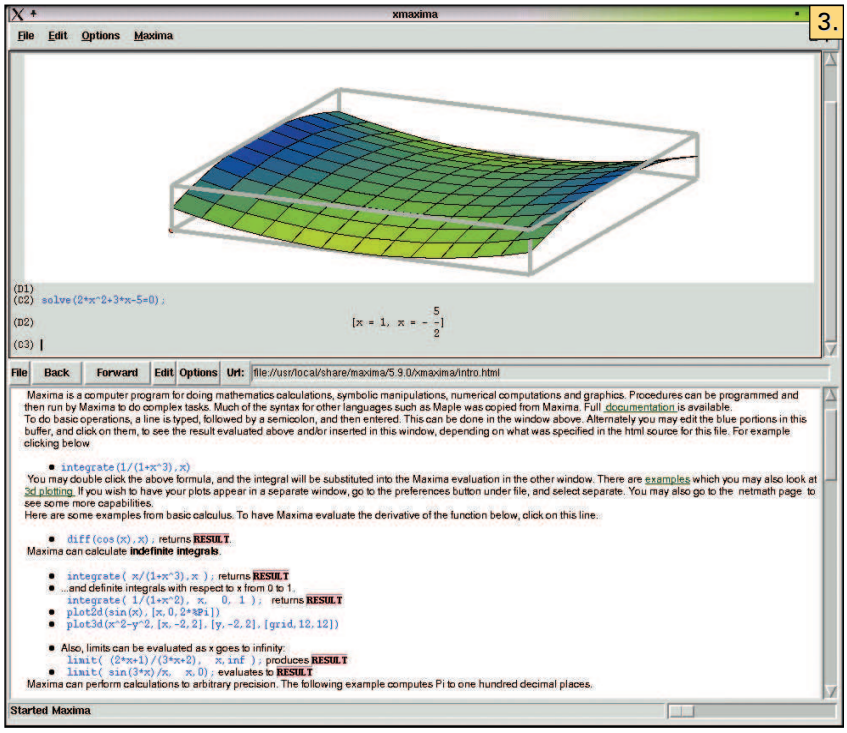
1.


2.

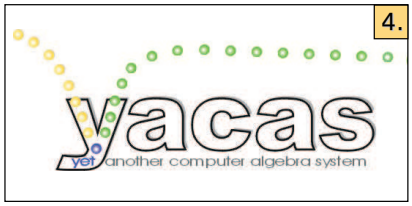
A Maple mellett többen használnak Mathematicát is. Mindkét profi rendszer komoly anyagi befektetést jelenthet számunkra, ezért magyar viszonylatban érdemes ezeknél esetleg „fapadosabb”, mégis jól használható programok után néznünk, különösen, ha sok felhasználó számára kell egyszerre elérhetővé tennünk ugyanazt a programot (például tanárként vagy rendszergazdaként).

Az első lehetőség a MuPAD, amit Németországban a Paderborni Egyetemen fejlesztettek ki, és a SuSE-terjesztés jó néhány éve tartalmazza. Legfrissebb változata a 2.5.2-es, amelyben korszerű háromdimenziós megjelenítéssel dolgozhatunk (*1. kép*). A ***Help/Help Documents/MuPAD Tutorial*** menüpont alatt TeX alapú, rengeteg példát tartalmazó hypertextes tankönyvet találunk, melyben a „2. First Steps in MuPAD” fejezetet célszerű először elolvasni.

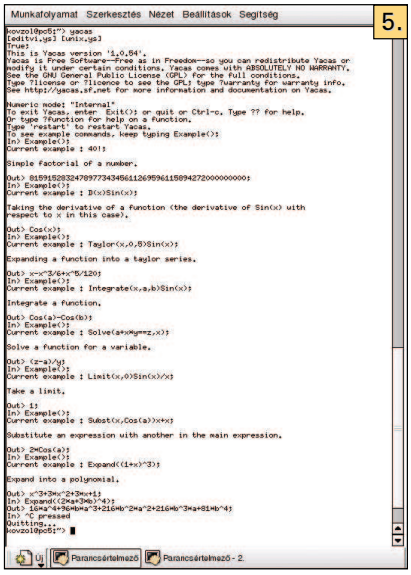
A MuPAD nyilvános változata memória-



3.



4.



5.


6.

korlátos (azaz bizonyos nehézségű feladatoknál visszautasítja a válaszadást), de ez a legtöbb felhasználó számára egyáltalán nem érzékelhető. Zárt forráskódja ellenére a MuPAD előnyös választási lehetőség mind a Maple, mind a Mathematica mellett, és mivel ingyenesen is hozzáférhető, az oktatásban is kezd elterjedni.

Léteznek más programok is, amelyek szabad forráskódúak:

- Az egyik a Maxima (➲ http://maxima.sf.net), ez az 1965-1972 között, az MIT-n fejlesztett legelső CAS, a Macsyma (➲ http://www.scientek.com/macsyma, 2. *kép*) szabad változata. Nemrégiben jelent meg az 5.9.0-s változat, ami a Linux mellett más operációs rendszereken is elérhető (*3. kép*).
- Egy másik friss fejlesztés a Yacas (➲ http://yacas.sf.net), ami a Yet Another CAS szavak rövidítése (*4–5. kép*). Az `Examples();` parancssor beírásával újabb és újabb példákat kapunk, amelyek segítségével elsajátítható a program alapszintű kezelése.
- Harmadik lehetőségünk az, hogy a GiNaC C++ eljárásgyűjteményt használjuk (*6. kép*), közvetlen módon hívva a `ginsh` matematikai parancsértelmezőt. A GiNaC-ot (➲ http://www.ginac.de) a németországi Mainz Johannes Gutenberg egyetemén fejlesztik, s kevesebbet tud ugyan más CAS-programoknál, viszont előnye, hogy C++-ban igen könnyű szimbolikus számításokat végezni a segítségével. (A *listánkon* látható programot a GiNaC info oldalán található példaprogram alapján percek alatt készítettem el, melyet a `g++ fÆjlnØv.cc -o kimenet -lcln -lginac` parancssorral fordítottam le futtatható kimenetté.)

Pillanatok alatt elkészíthető, GiNaC alapú programocska

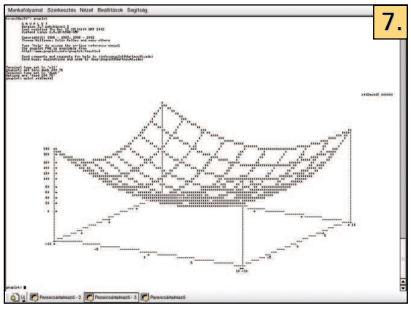
```
#include <iostream>
#include <ginac/ginac.h>
using namespace std;
using namespace GiNaC;

int main()
{
  symbol x("x"), y("y");
  ex poly;

  for (int i=0; i<10; ++i)
    poly += factorial(i)
    ➥*pow(x,i)*pow(y,10-i);

  cout << poly << lendl;
  return 0;
}
```

A Maxima LISP nyelven íródott, a Yacas és a GiNaC pedig C++-ban. Mindhárom program mögött komoly fejlesztői szervezet áll, azaz hosszú távon a három közül bármelyik program vetélytársa lehet a kereskedelmi komputeralgebra-rendszereknek.

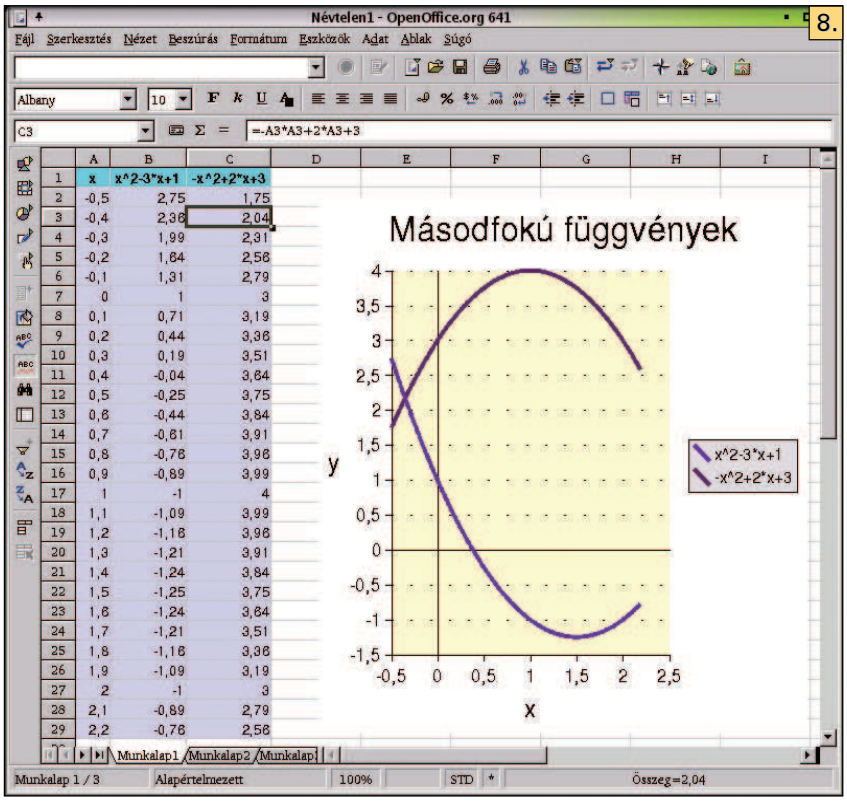
Egyelőre nem említhető egy lapon a fenti programokkal, mégis hadd számoljak be egy olyan munkáról is, amelyet kollégáimmal kezdtünk el a Szegedi Tudományegyetemen. A WebMathematics Interactive nevű, számos szabad programra épülő fejlesztésünk – meglévő CAS-programok támogatásával – középiskolások, főiskolások, egyetemisták számára ad lehetőséget gyakori matematikai feladatok számítógépes megoldására a saját anyanyelvükön. A diák ellenőrizheti is a tudását a folyamatosan bővített tesztkérdés-adatbázis segítségével. Programunk (GPL felhasználási szerződésű) a ➲ http://pascal.math.u-szeged.hu/wmi címen próbálható ki.

Függvényábrázolás, geometria, fraktálok

A teljesség igénye nélkül lássunk pár példát más jellegű matematikai programokra is! A legismertebb Unix alapú függvényábrázoló a `gnuplot`, aminek nemrég jelent meg a 3.7.3-as változata.

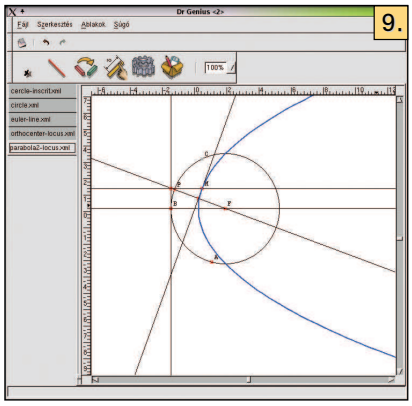

7.

szerkesztési feladatok megoldásánál nagy örömet szerezhetnek az úgynevezett bázispontok felvételén alapuló geometriai szerkesztőprogramok. Minden megszerkesztett objektum (szakaszok, körök) előre adott bázispontokra épülnek, és a bázispontok későbbi mozgatásakor az elkészült rajz teljes egészében dinamikusan újrarajzolódik. Két olyan szerkesztőprogram is rendelkezésre áll, ami a magyar nyelvet is



8.

Ez a program igazán klasszikusnak számít a Unix-történetben, hiszen 1984 óta fejlesztik. Előnye, hogy interaktív módon kezelhető, s kimenete számos előre beprogramozott formátum lehet (például: PostScript, X-ablak, vagy akár ASCII-jelekből készített blokkgrafika, mint az a *7. képen* is látható). Hátránya, hogy nem eléggé felhasználóbarát, vagy inkább fogalmazzunk úgy: kezelőfelülete nem mindenki számára kényelmes. Vannak más, grafikus kezelőfelületű függvényábrázolók is, melyek azonban a kezdeti fejlesztés után rendszerint elakadtak. Éppen ezért ha Linux alatt, „korszerű” módszerek segítségével szeretnénk a gyerekekkel a függvényábrázolást gyakorolni, akkor – mivel a táblázatkezelés úgyis iskolai tananyag – kipróbálhatjuk például az OpenOffice.org beépített táblázatkezelőjét (*8. kép*).
Az elemi geometria tanításában, a

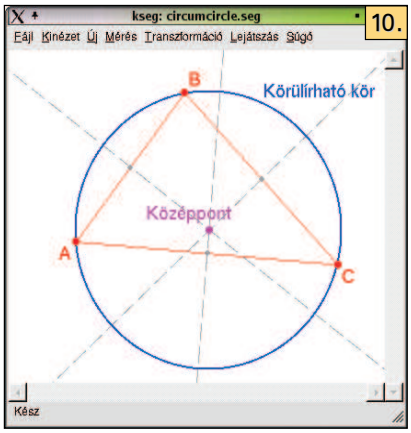

9.

támogatja. Az egyik a DrGenius (*9. kép*), amelyet elődje, a DrGeo folytatásaként fejlesztenek programozó tanárok. Ez a program kevés szöveggel, számos beszédes ikonnal intuitív módon kezelhető. A másik a `kseg` (*10–11. kép*); különlegessége a szerkesztési lépések egymás-

utánjának a kijelzése. Alapértelmezésben a `kseg` program angol nyelvű, ezért a ***File/Choose Language…*** menüpontjában keressük meg és válasszuk ki a ***kseg_hu.qm*** nevű fájlt (ez kimásolható a forrás programcsomagjából, ha nem lenne fenn a gépünkön). Mindkét geometriai program támogatja az elkészített ábrák PostScript formátumú mentését is, ami a közvetlen kinyomtatás mellett további feldolgozásukat segítheti elő.
A matematika megszerettetésében nagy szerepük lehet a látványon alapuló programoknak. Sok olyan emberrel találkozom nap mint nap, akik ódzkodnak a matematikától, a különleges matematikai ábrák láttán megdöbbennek, ám azok mégis felkeltik az érdeklődésüket. Bár a `surf` nevű program kezelése nem egyszerű, az elkészített háromdimenziós képek szépsége kárpótol bennünket a fáradságért (*12. kép*). Több olyan program is létezik, amivel fraktálképeket állíthatunk elő. A KDE beépített fraktálkészítője (a `kfract`, *13. kép*) könnyen kezelhető, ám például a XaoS-hoz (*14. kép*) viszonyítva kevés támogatást nyújtó program. Ez utóbbival viszont – amellett, hogy a legfrissebb változat 13 különböző fraktáltípust ismer – valós időben, mozifilmszerűen nagyíthatunk bele az egyes fraktálokba.

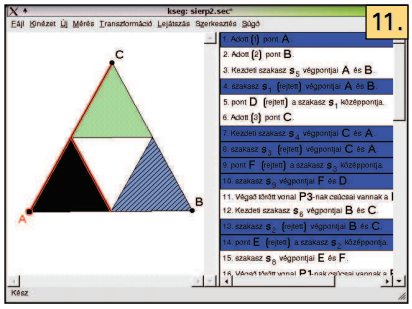
Szövegszerkesztés, matematikai képletek

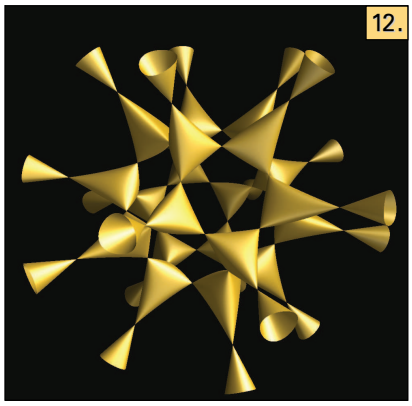
Ha matematikai szöveget írunk Linux alatt, legkézenfekvőbb eszközként valószínűleg a TeX program jut először eszünkbe. Én sikerrel használom a LyX-et, amely LaTeX alapú, és azonnal

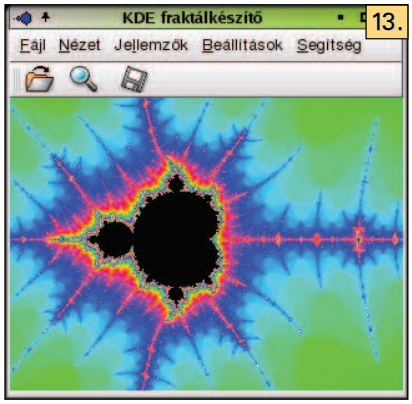


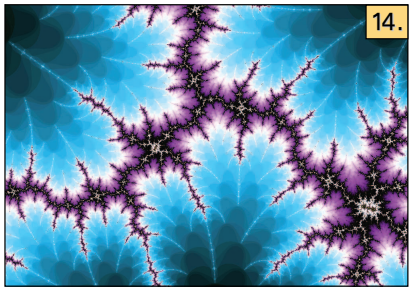
10.

képes akár PDF-, akár PostScript- vagy HTML-kimenetet is létrehozni (ez utóbbit a `latex2html` átalakítóprogram segítségével). A legutóbbi változat már többféle képformátumot is be tud tölteni, így nagyon kényelmesen és gyor-

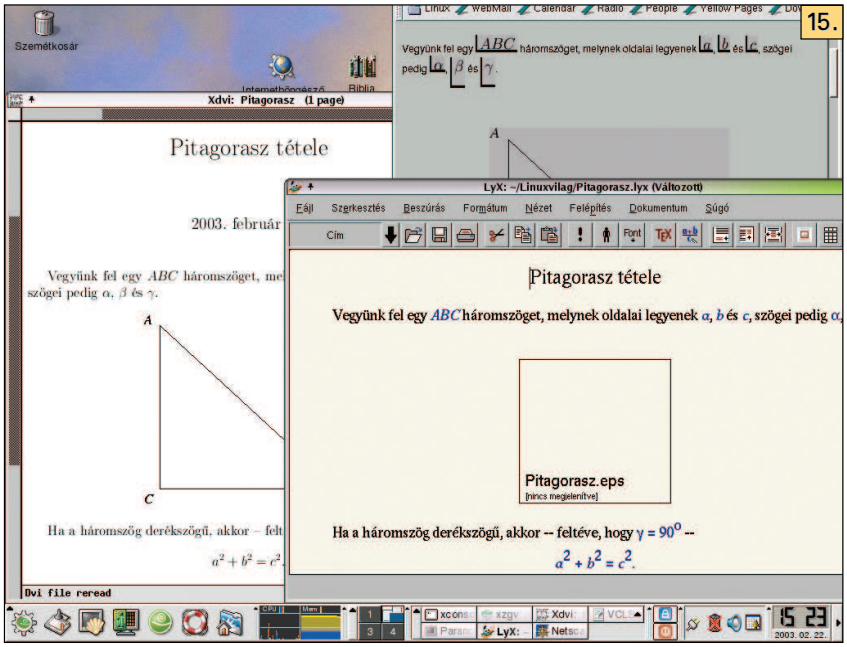


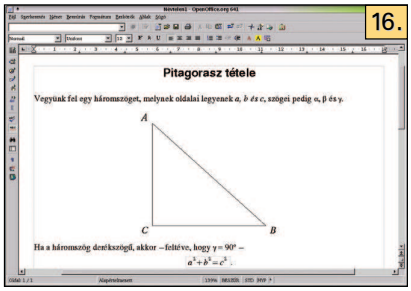






san profi minőségű szöveget elő tudunk vele állítani. Ha a nyomtatás minősége fontos, akkor feltétlenül PostScript formátumban érdemes elkészíteni a dokumentum ábráit, például az `xfig` programmal; ez azonban sokszor nem kis szakértelmet kíván. Tapasztalatom szerint a legkönnyebben StarOffice-szal készíthetünk szép, pontos PostsScript-ábrákat, amelyeket könnyen át tudunk





17.

2. dolgozat: Egyenletek, szöveges feladatok

B csoport

1) $2\cdot\frac{x}{3}-\frac{x}{4}=5$

2) $\frac{x+3}{2}=\frac{3x-5}{6}$

3) $4\cdot(x-5)\geq 3\cdot(x-5)-(2-10x)$

4) Egy szám a másiknak ötszöröse, különbségük 32. Melyik ez a két szám?

vinni LyX-be is. Az ábra elkészítése után jelöljük ki a teljes ábrát (pl.: CTRL-A-val), majd a ***File/Export…*** menüpont alatt ***.eps*** formátumban mentsük, ügyelve, hogy a ***Selection*** lehetőség be legyen jelölve. A mellékelt képen (*15. kép*) láthatjuk, hogy a LyX-szel készült szövegből ***.dvi*** kimenet és ***.html*** is készíthető (a gépemen futó `latex2html` túl régi, ezért a ***.html*** kimenet nem túl szép; a legfrissebb változat sokkal látványosabb dokumentumot gyártana).

Van, akinek a TeX/LaTeX alapú rendszerek túlságosan bonyolultak. A StarOffice, illetve OpenOffice programok első látásra könnyebben kezelhetőnek tűnnek, mint a LyX vagy a TeX/LaTeX. Azonban sok apró trükkre rá kell jönnünk, amíg ténylegesen úgy néz ki a dokumentumunk, ahogyan szeretnénk, és kezdetben egészen egyszerű matematikai szövegeket is csak hosszas munkával tudunk létrehozni. A feleségem, aki szintén matematika szakos tanár, másfél év alatt komoly tapasztalatot szerzett a StarOffice-os képletszerkesztő használatában, és – velem ellentétben – nagyon gyorsan bonyolult feladatsorokat is össze tud állítani a segítségével. (A *16. képen* a saját munkám, a *17-en* a feleségemé látható.)

Zárszó

A fentiekben épp csak be tudtunk pillantani a linuxos matematikai programok némelyikébe – rajtuk kívül sok olyan program létezik, ami említésre méltó és a gyakorlatban is sikerrel alkalmazható. A programok használati utasításai, a kézikönyv- és infooldalak, továbbá az internetes források további útmutatást adnak az érdeklődő olvasók számára.

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.

Xandros Desktop 1.0

A Linux fejlődése afelé mutat, hogy előbb-utóbb meg fog szűnni a Microsoft egyeduralma a munkaállomások operációs rendszerének a piacán. Legalábbis remélem, hogy ez a változás minél hamarabb be fog következni.

Számos olyan projekt indult el, ami megváltoztatta az egyszerű felhasználók hozzáállását a Linuxhoz, és igenis bebizonyította, hogy létezik olyan választási lehetőség, ami képes a megszokások emberét egy másféle, mégis hasonló operációs rendszerhez átcsábítani.
Kezdetben volt a Progeny, ami – hasonlóan a Xandroshoz – Debian-alapokon nyugvó operációs rendszer volt, illetve nemcsak volt, hanem az is, csak a fejlesztése állt le.
Tavaly november tájékán jelent meg a Xandros, ami Corel Linux-örökséggel bír. Bevallom, nagyon vártam, hogy mikor jutok már hozzá egy ilyen operációs rendszerhez, mivel a Corel igazi áttörést jelentett számomra. Mindazt tudta, amit a „nagyok" nem, és mivel a Corel elsősorban grafikai programjairól híres, ez nagymértékben meglátszott a KDE felületén. A nagy várakozásnak vége, itt a Xandros Desktop 1.0 Debian 3.0-s alapokon.
A termék a Xandros Desktop 1.0 CD-jét, a KD3 technológiai előzetesét (csak erős idegzetűeknek, a Xandros ipari felhasználás esetén nem ajánlja a használatát), valamint a nyomtatott leírást (250 oldal) tartalmazza. Továbbá jár hozzá a 30 napos terméktámogatás, egyebekről nem is beszélve (például regisztrációs kártya).

A telepítésre szánt eszköz:
- HP Vectra VL420 DT
- iiyama LCD monitor
- Intel PRO/VE 100 hálózati kártya
- Intel 810 beépített hang

A telepítőprogram a Corel Linuxhoz hasonlóan nagyon egyszerű, sőt szerintem ennél könnyebben használható már nem is lehetne. Az üdvözlőképernyővel és a felhasználási szerződést tartalmazó ablakkal együtt összesen 12 képernyőn kell végiglépkedni ahhoz, hogy a Xandros legyen az operációs rendszer a gépünkön. Érdemes azért végignézni, hogy milyen kérdéseket tesz fel a telepítő!

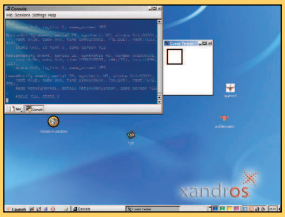
Rendszerjellemzők

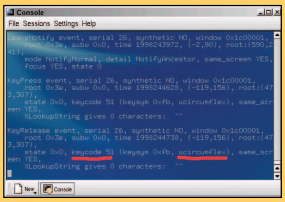
	Minimum	Ajánlott
Processzor	Intel Pentium vagy Celeron processzor, AMD K6/II/III, Duron, Athlon, Athlon XP/MP processzor	Pentium II 450 MHz vagy AMD K6/II
RAM	64 MB	128 MB
Merevlemez	1,5 GB szabad tárhely	3 GB szabad tárhely
CD-ROM	Igen	Igen
Hajlékonylemez-meghajtó	Csak akkor szükséges, ha a gép nem tud CD-ről rendszert tölteni	Csak akkor szükséges, ha a gép nem tud CD-ről rendszert tölteni

A Xandros szerint szükséges eszközök ***táblázatunkban*** láthatók.
A felhasználási szerződés egy kereskedelmi gépre, illetve korlátlan otthoni, illetve nem kereskedelmi gépre történő telepítést engedélyez. Ez annál is furcsább, mert a Xandros honlapján (➲ http://www.xandros.com) egyértelműen az olvasható, hogy a teljes Desktop változat ára 99 dollár, és ez a tétel nem más, mint együtt a CodeWeavers CrossOver Office és a CrossOver Plugin összesen 69, plusz a Xandros 30 dollárja. A Standard Desktop pedig 39,95 dollárt kóstál; hogy mitől kerül többe tíz dollárral a butított változat, az kérdés marad, mivel minden különleges szolgáltatás hiányzik belőle (NTFS lemezrész átméretezése, CodeWeavers programok), és nincs hozzá csak egyetlen alkalomra szóló terméktámogatás. A tesztek alapján sokkal inkább tudom ajánlani a Custom telepítését, mert az Express üres merevlemezre ajánlott.
A telepítő csak angolul szólal meg, és a kész telepített rendszer is csak kevés nyelvet támogat (a magyart nem), igaz, némi felkészültséggel és kitartással sokat lehet javítani a Xandros nyelvi támogatásán.
A telepítési folyamat után az első indításkor a Xandros fekete képernyője a pulzáló Xandros logóval fogad. Puritánabb, mint a SuSE színes animációi, mégis sokkal elegánsabb. Néhány üzenet és a KDM grafikus beléptető program köszön ránk.
Az első grafikus indítást követően önműködően elindul a ***First Run Wizard***.

Hogyan készítsünk Xmodmap fájlt?

A módszer ősrégi, de szerintem a legbiztosabb eredményt nyújtja. Miről is van szó? Az `xmodmap` parancs segítségével olyan billentyűzetkiosztást lehet betölteni, amit a felhasználó szeretne.





Mire van szükség ehhez? Először is szükség van egy `xterm` ablakra, majd ott ki kell adni az `xev` parancsot. Megnyílik egy kis ablak, és megváltozik az `xterm` tartalma is. Fontos, hogy az egérrel ki kell jelölni a kis fehér ablakot, vagyis annak az üres területét, és akkor kell lenyomni a billentyűt. A lenyomott billentyű kódja megjelenik. Két fontos adatra lesz szükség az Xmodmap fájl megírásához, a keycode számra, jelen esetben ez a keycode 51, valamint a zárójelben lévő utolsó adatra, példánkból kiindulva az ucircumflexre. Ezt a folyamatot, eljárást minden olyan karakterre, illetve billentyűre el kell végezni, ami nem megfelelően jelenik meg a képernyőn vagy írás közben a szövegben. Ha minden hibás tétel megtaláltatott, akkor el kell készíteni az Xmodmap fájlt. A fájl helye a */etc/X11/Xmodmap*, általában minden Linux-terjesztés tartalmaz egy ilyet.

Szerkezete a következőképpen fest:

keycode szám = <kisbetű kódja> <a shift lenyomására mi jelenjen meg> <az Alt Gr lenyomására mi jelenjen meg>

Példánkból következik:

```
keycode 51 = ucircumflex Ucircumflex
```

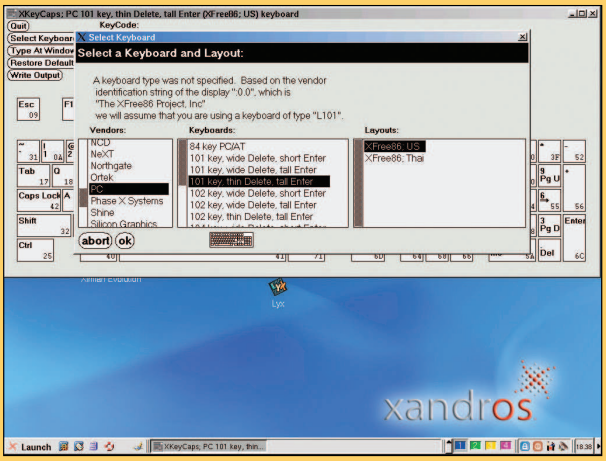
Itt az ű Ű fog megjelenni, és mivel az ALT GR lenyomásával adott karakterre nincs szükség, ezt nem kódoltam le (a fájl a 46. CD Magazin/Xandros könyvtárában található).

Létezik egy másik lehetőség is az Xmodmap létrehozására, ez pedig nem más, mint az xkeycaps program, ami a 46. CD Magazin/Xandros könyvtárában található, vagy a ➲ http://www.jwz.org/xkeycaps honlapon érhető el. Az xkeycaps telepítése nagyon egyszerű, a következő lépésekre van szükség:

1. Ki kell csomagolni a *tar.z* fájlt.
2. Adjuk ki az `xmkmf` utasítást. Itt akadt egy kis hiba: a Xandros nem tartalmazta a szükséges állományokat, ezért le kellett tölteni a libxaw6-dev és a libxaw7-dev csomagokat (természetesen a libxaw6 és libxaw7 is szükséges, de ezek a csomagok fenn voltak a rendszeren).
3. `make`
4. `make install`
5. `make install.man`

A telepítést követően `xkeycaps` paranccsal lehet meghívni a programot és beállítani a billentyűzetkiosztását.

A program alapból a *../home* könyvtárba menti az Xmodmap fájlt, ekkor azonban csak az adott felhasználó képes az Xmodmap fájl használatára, ha tehát valamennyi felhasználónak ugyanazt a kiosztást szeretnék kiadni, akkor a fájlt be kell másolni a */etc/X11* könyvtárba.



A telepítés

A varázsló végigvezeti a felhasználót mindazokon a lépéseken, amelyek kiindulópontul szolgálnak a munkaasztal beállításához, valamint a nyomtató beállítása is megtörténik. A nyomtató beállításánál hiányzott, hogy nem láttam nyomtatókereső gombot. Többek között azért is igényeltem volna, mert ez a szolgáltatás a Mandrake-ben benne van.

A rövid újraindítás után fedezzük fel a Xandros nyújtotta pluszokat, újdonságokat! Engem legfőképp a Xandros Networks szolgáltatás érdekelt. Mit tud nyújtani, illetve mennyire képes kiszolgálni a felhasználói igényeket? Keserű csalódás, hogy a programkínálat enyhén szólva is szegényes, a frissítések sem jöttek olyan ütemben, mint ahogy az elvárható lenne.

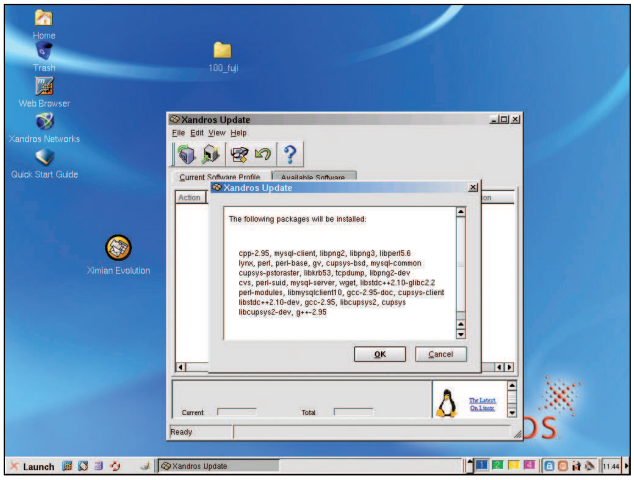
Az első telepítéskor semmilyen biztonsági frissítés nem volt, decemberben kezdett a számuk gyarapodni és akkortájt kezdték kijavítani a felhasználók által jelzett hibákat.

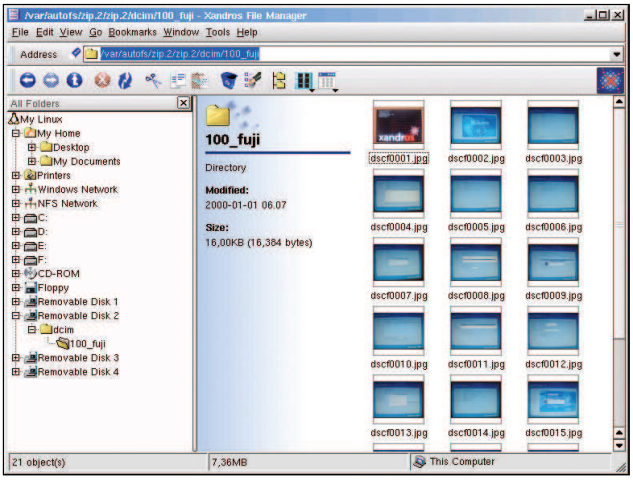
Frissítések

A következő csoportok találhatók a Xandros Networks-frissítések között:

- Alkalmazásfrissítések: kanadai francia billentyűzetkiosztás, internetcsatlakozás-frissítés, Apache-frissítés, CodeWeavers CrossOver Office-frissítés.
- Biztonsági frissítések: KDE biztonsági frissítés, Xandros Networks biztonsági frissítés, és a sokak által hiányolt tűzfalprogram (alapból ugyanis nincs a rendszerben), a Firestarter.
- Kritikus frissítés: a Xanadros Dekstop eszközfelismerő rendszer frissítése.
- Ajánlott frissítések: windowsos hálózatkeresés-frissítés, a Linux által `ext2` fájlrendszerre formázott meghajtók olvasási frissítése.
- Meghajtóprogram-frissítések: ATI Radeon 8500-9700, Aureal Vortex AU8810 és AU8820, eszközkereső alrendszer az Intel 810-hez, HP Office Jet Linux Driver, HP Linux inkjet és Foomatic, Tekram modulfrissítés, Matrox kereskedelmi meghajtó.
- Nyomtatásfrissítés: a Lexmark, HP Laserjet, Canon és Epson nyomtatókhoz.
- Kísérleti frissítés: ALSA hangkártya-meghajtók.

Furcsa kettősség jellemzi a Xandros frissítési és csomagkezelési programjait. Elsősorban a munkaasztalon szereplő ikonnal indítható a Xandros Networks és a Corel örökségből jött a Xandros Update program. Miután a Xandros Networksről minden frissítést leszedtem és megnyugodtam, hogy most aztán minden tökéletes, hibátlan az operációs rendszerem,





elképedve tapasztaltam, hogy a Xandros Update indításával olyan mennyiségű csomagtenger ömlött elém, ami egy kicsit hervasztó volt.

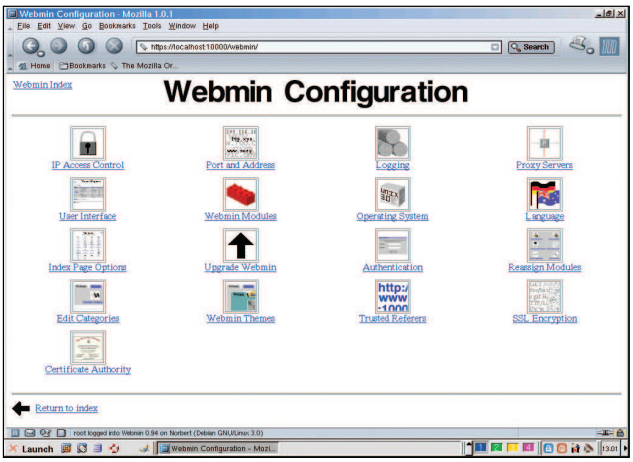
Sokkal átgondoltabbnak tartottam volna, ha frissítéseket az operációs rendszer egy helyen kezeli, mert végül is a Xandros Networks esetében is van ***Updates*** bejegyzés. Tovább rontja a helyzetet, hogy bár a Xandros Debian 3.0-s alapokon nyugszik, csak a csomagforrások bekapcsolása után érhetőek el a Debian 3.0-ra vonatkozó frissítések – mi több, a frissítések sorának még mindig nincs vége, mert az előbbi esetből okulva ismét meglátogattam a Xandros frissítési oldalát, ahol újabb frissítésre kijelölt csomagokat tudtam leszedni.

A csomagforrás kiegészítésével azonban hozzájuthatunk a Debian nyújtotta teljes programkínálathoz, amiből kedvünkre válogathatunk, mivel a Xandros Networksnek megvan az a jó tulajdonsága, hogy kitűnően kezeli a Debian csomagfüggőségeit. Végül is erre van szükségük a felhasználóknak, azokat a programokat válogatja le és teszi fel, amire szüksége van. Az operációs rendszer hordozhatóságához hozzájárul az is, hogy kitűnő az alkatrész-felismerési aránya és eszköztára, például a Fuji fényképezőgép memóriakártyáját egy olyan kártyaolvasóval másoltam le, ami Windows XP alatt csak meghajtótelepítés után volt hajlandó működni, valamint összekeverte a teljes meghajtólistát; ezzel szemben a Xandros, amint csatlakoztattam az eszközt, azt máris megjelenítette, s a kártya tartalma is azonnal elérhetővé vált.

Mivel a munkám során hozzászoktam a magyar karakterekhez, megpróbáltam a Xandrost legalább arányaiban elmozdítani a magyar nyelvi támogatás felé, olyan eszközöket keresve, amik – elsősorban – a kezdő felhasználók kedvét nem veszik el a kísérletezéstől. A Xandros Networksön rákerestem a hungarian szóra, az eredményt lásd a képen.

A ***hunglish*** csomagot érdemes telepíteni, (a ***hunglish-x3***-mat még véletlenül se tegyük fel, mert tönkreteszi a ***Control Center***-t). Vannak itt még betűtípusok, ***ispell***, ***myspell*** és a ***kde-i18n-hun*** csomag, amivel ugyan részben, de magyarítani lehet a KDE felületét. Azért csak részben, mert azok a szolgáltatások, amelyeket a Xandros ültetett be a KDE-be, nem érhetők el magyar nyelven. Be kell vallanom, én nem állítottam át a KDE alapnyelvét magyarra, sokkal inkább a részletek érdekeltek. Mivel a ***hunglish*** csomagot feltelepítettem, azt vártam, hogy most végre lesz magyar nyelvű billentyűzetkiosztás, de sajnos tévedtem.

A legszebb az egészben az volt, hogy míg a Mozillában vígan írhattam a magyar karaktereket, addig a Kwrite-ban nem. Egy ősrégi módszerhez nyúltam, és a még Red Hat 6.2-höz írt



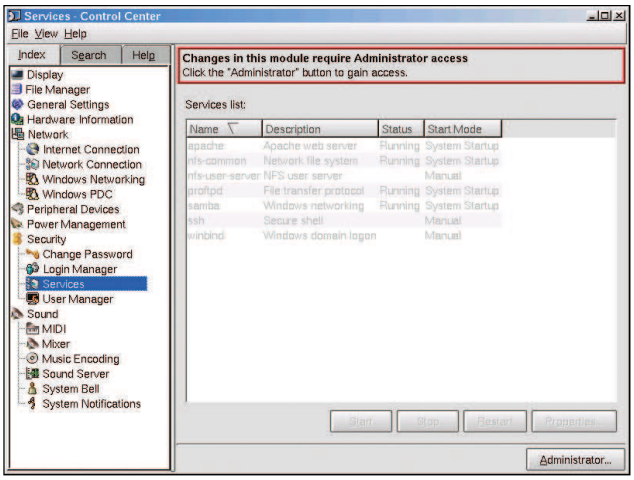
Xmodmap fájlt másoltam be a */etc/X11* könyvtárba, felülírva az ott lévőt. Láss csodát, lett mindjárt magyar karakter, billentyűzetkiosztás, és minden a helyére került; igaz, annyi finomítás még elkelt, hogy az `xterm` betűtípusát át kellett állítani Unicode-ra.

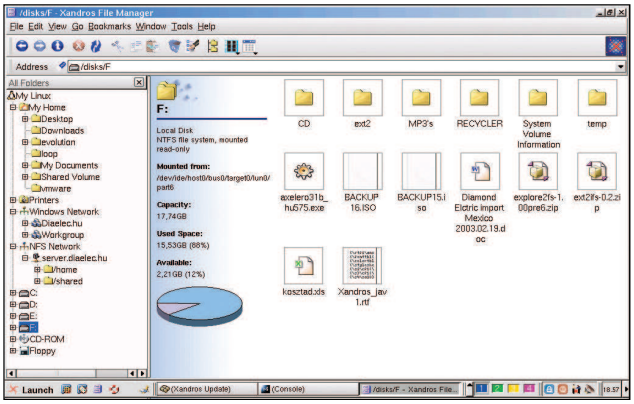
A Xandros beállítása

A Xandros beállításaival folytattam a munkát. Az egyéb Linux-változatokhoz képest a ***Control Center***-en kívül alaphelyzetben semmilyen grafikus beállítóprogram nem található, tehát irány a Xandros Networks, és szedjük le a linxconf és Webmin csomagokat. A Webmin csomag a Xandros Networksön olyan ősrégi, hogy azonnal frissítenem kellett a ➲ http://www.webmin.com oldalról, a Webmin belső frissítési lehetőségét kihasználva. Így megkíméltem magam a Webmin moduljainak letöltögetésétől is, mivel a Webmin belső frissítése önműködően letölti a tarfájlt, majd telepíti az összes elérhető legfrissebb modult.

Ugyanezzel a lendülettel szedtem le az Squidet, és a Webmin segítségével be is állítottam. Fontos megjegyezni, hogy a Xandros Networks proxyn keresztül nem működik, illetve az esetek többségében nem – hol működik, hol meg nem, tehát arra az időre ki kell kapcsolni a Mozillában a proxybeállításokat, amíg a Xandros Networksöt használjuk.

Bizonyos beállításokat ugyan tudunk végezni a KDE ***Control Center***-ből is, ahol minden esetben piros keret figyelmeztet, hogy rendszergazdai joggal kell belépni a beállításokhoz, amit az ***Administrator*** gombra kattintva meg is tehetünk.





Ha mindezt alapértelmezésben tartalmazta volna a rendszer, megkímélt volna a kellemetlenségektől. Jó lett volna, ha benne lett volna a telepített Xandrosban (mert igazából arról kellett volna meggyőznie a Xandrosnak, hogy érdemes a rendszerét feltenni a Windows helyett), de nem ez volt az első meglepetés, amit a Xandros okozott.
Miután mindenféle beállítóeszközzel felvérteztem magam, felmértem, mit is tud nyújtani a programok terén. Nem vagyok híve azoknak a Linux-terjesztéseknek, amelyek 5–7 CD tartalmát ölelik fel, és az egyszerű felhasználó a programok mintegy 70–80 százalékát nem használja. A Xandros valahol az egészséges használhatóság határán van, mindazokat az alapprogramokat tartalmazza, amikre egy munkaállomáson szükség van: beépített OpenOffice-t, Kiviót, Acrobat Readert, RealPlayert, és említést érdemel még a Xandros Networksről letölthető Learning Guides is.
Választási lehetőségként ott van a Xandros Networks, ami olyan mennyiségű programot tartalmaz a Debian-források bekapcsolását követően, hogy az már-már több mint elég. Gondok akkor kezdődtek, amikor a programot forrásból kellett fordítani, igaz, az egyszerű felhasználó ilyet nem tesz, de akadnak olyan programok, például a VMware, amelyeket csak így lehet telepíteni.
A rendszermagforrást tartalmazó fájl fenn van a */usr/src*-ben, de nincs kicsomagolva. Ezen könnyen lehet segíteni:

```
tar jxf xandros-kernel-source-2.4.19.tar.bz2
```

De ezzel sajnos még nincs vége.
További nehézség, hogy a VMware telepítője a használt rendszermagot nem találja egyezőnek, valamint a felrakott és kicsomagolt forrást. Megoldás:

```
1. cd /usr/src
2. ln -sf xandros-kernel-source-2.4.19 linux
3. cd linux
4. make EXTRAVERSION=-x1 mrproper
5. cp /boot/config-2.4.19-x1 .config
6. make EXTRAVERSION=-x1 oldconfig
7. make EXTRAVERSION=-x1 dep
8. cd /usr/src
```

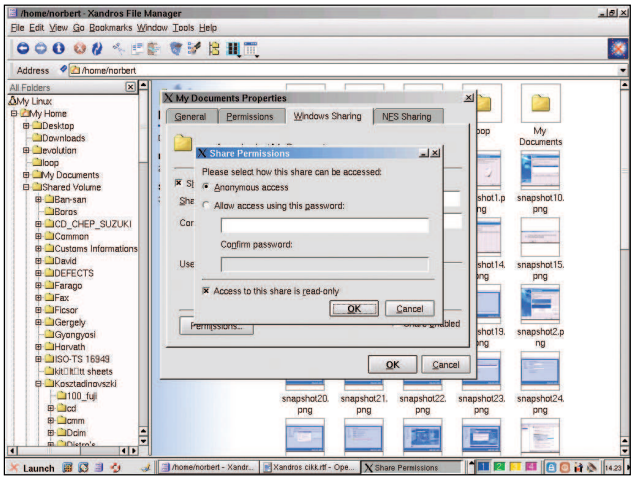
Majd jöhet a `vmware-install.pl`, ezt követően a `vmware-config.pl`, s ha mindez megvan, akkor a */etc/devfs/devices* fájlhoz hozzá kell adni a következő sorokat:

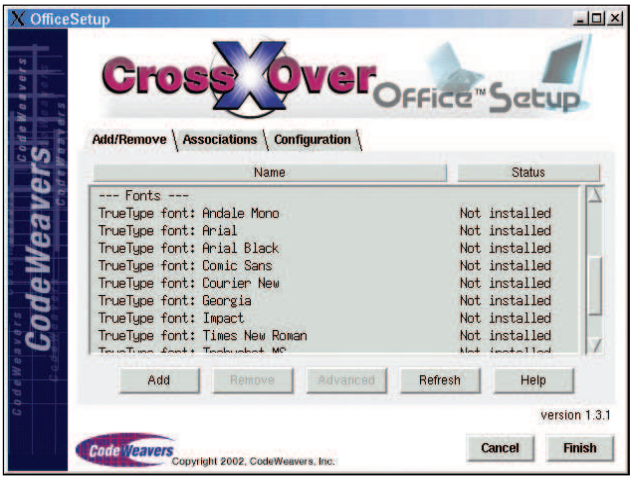
```
vmnet0 c 119 0 0 0 0660
vmnet1 c 119 1 0 0 0660
vmnet2 c 119 2 0 0 0660
vmnet8 c 119 8 0 0 0660
vmnet9 c 119 9 0 0 0660
```

Ellenkező esetben minden újraindítást követően elvesznek a VMware-modulok beállításai, és újra lehet futtatni a `vmware-config.pl` parancsot!
Így most már működik a VMware, és feltehetőleg a többi forrásból fordított program esetében sem lesz gond a fordítás és telepítés folyamatával.
Lenyűgöző a Xandros fájlkezelője: minden elérhető létező meghajtót és eszközt kezel, mi több, önműködően fűz be Samba-meghajtókat. Sőt hibátlanul kezeli a Windows NTFS fájlrendszerét is, s a telepítő – a SuSE-hoz hasonlóan – az NTFS-kötetekhez a csatolásokat önműködően létre is hozza.
Azonban itt is felmerült egy nehézség: a magyar karakterek támogatása, amit azzal lehet megoldani, hogyha a */etc/samba/smb.conf* fájlban a `[global]` részt a következő sorokkal egészítjük ki:

```
client code page = 852
character set = ISO8859-2
```

Ez annyiban orvosolja a gondot, hogyha a fájlkezelő egy Samba-meghajtót önműködően befűz egy ideiglenes könyvtárba, akkor megjelenik az őŐűŰ és társai, míg ha a fájlkezelő ***Tools Mount network share*** programot futtatjuk, akkor továbbra sem jelennek meg a magyar ékezetes karakterek. Viszont





mindez a kellemetlenség feledteti azt a sok lehetőséget, amit a fájlkezelő nyújt. Bármit bárhova másolhatunk, kicsomagolhatunk, mert minden eszköz automatikusan elérhető, ugyanez igaz az általunk megosztani kívánt könyvtárakra is.
Mesteri, szinte hihetetlen megoldás, hogy most találkoztam az első olyan Linuxszal, amiben a kuka ugyanúgy működik, mint a Windowsban, vagyis az ott tárolt törölt elemek bármikor visszaállíthatók az eredeti helyükre – ezt eddig egyetlen nagy Linux-kiadás sem tudta.
Ugyanilyen fontos, hogy a Xandrosban megtalálható egy a Windowshoz hasonló számítógépkereső program is.
Természetesen ha már munkaállomásokról beszélünk, fontos lehet a játék, illetve a játékprogramok futtatása is. A Xandros CD csak a KDE-alapjátékokat tartalmazza, amit ki lehet bővíteni a Xandros Networks által nyújtott szintén csekély számú programmal, ezt követően viszont érdemes a Xandros Networksöt ***Expert*** módba átkapcsolni, és a megnyíló könyvtárszerkezetben bőven van Debian 3.0-s játék. Sajnos a gépem, illetve a Xandros ismét kihívásokkal küszködött, mert az Ati Rage PRO 128 GL-nek esze ágában sem volt futtatni bármit is. Pedig a Xandros tökéletesen megtalálta a kártyát és telepítette is, sőt az ***XfreeConfig-4***-ben is szerepel a helyes r128 meghajtó, valamint a ***Load glx*** bejegyzés. Vajon mi a megoldás?
A Xandros nem telepítette az alábbiakat:

```
/disks/Removable/cdrom.1/dists/xandros1.0/main
/binary-i386/base/xandros-r128agp-drm-modules-
2.4.19-x1_0.0.2002.09.25.15.45.x.1.0-
3_i386.deb
```

A csomag felrakása és egy újraindítást követően végre „emberi” sebességgel és minőségben futnak a játékok is. Mit is lehetne még megemlíteni? Talán két számomra kedves és fontos dolgot! Elsőnek a „switch user” programról szólnék, ami a tálca jobb sarkában bújik meg.
A program képes arra, hogy egy másik felhasználóra váltson át, és egy új KDM-belépéssel egy új grafikus munkaasztalt nyisson meg, anélkül, hogy a kezdeti kiinduló munkaasztalon bármi változna. A munkaasztalok között a CTRL+ALT+F1 billentyűk lenyomásával lehet váltogatni. A másik érdekes és a Windowshoz szokott közönség érdeklődését felkeltő program a CodeWeavers CrossOver Plugin és CrossOver Office.
A Plugin képessé teszi a KDE-t, valamint a Mozillát arra, hogy az átírt Wine segítségével windowsos programokat, illetve azok beépülő moduljait tudjuk futtatni.

A beépülő modulok között megtalálható az Apple QuickTime6, de a Windows MediaPlayer 6.4 is. Jellemzően mind a Plugin, mind az Office tartalmaz betűtípusokat is, amelyek a futtatandó windowsos programok tökéletesebb megjelenítését teszik lehetővé. Az Office többek között lehetővé teszi a Microsoft Office, az Office 200, a Lotus Notes, a Visio futtatását, de itt található az Internet Explorer 5.5 is, valamint a TrueType betűtípusok.
Hogy is néz ez ki a gyakorlatban? A CrossOver Office futtatását követően például kiválasztjuk a QuickTime6-ot, majd az ***Add*** gombra kattintva elindul a telepítés.
Fontos megjegyezni, hogy a QuickTime6 esetében csak a minimális telepítést szabad futtatni, mivel a teljes telepítéskor nálam a CrossOver Office lefagyott. A telepítés befejezésével viszont a telepített ***dll***-fájlokat a Mozilla vígan használja, mintha azok a saját vagy a Netscape beépülő moduljai lennének.
Mindent egybevetve a Xandros még nem teljesen kiforrott, bár olyan tulajdonságokkal bír, amelyeket kevés Linux-terjesztés tudhat a magáénak. Számomra furcsa, hogy miért ragadtak le a fejlesztők a 2. változatú KDE-nél, szerintem sokkal nagyobb tábort tudhatna magáénak a Xandros, ha a fejlesztés már a KDE 3 alapokon folyt volna és nem csupán előzetesként szerepelne. Furcsa továbbá, hogy feltehetőleg pénzügyi okokból adták ki a Xandros Desktop Standard változatát, ami – mint azt már említettem – többe kerül, mintha levonjuk azon összetevők értékét, amelyeket kivettek belőle. Az sem igazán kedvez a Xandrosnak, hogy míg december végéig egy futárszolgálat ingyen hozta házhoz a megrendelt csomagot, addig mostanra ez is pluszköltséget jelent. Sajnos jelenleg egyetlen magyar értékesítője sincs a Xandrosnak.
Januárban megjelent a SuSE Linux Office Desktop. A SuSE több hozzáadott értéket préselt termékébe, többek között a CodeWeavers programok mellett a StarOffice 6-ot, valamint az Acronis OS Selector programot is. Ezt azonban magasabb, 129 dolláros áron forgalmazzák.
A verseny elindult, és ennek csak mi, felhasználók vagyunk a nyertesei. Vajon mi lesz ennek a folyamatnak a vége? Ki tudja! Kitartóan várom, hogy mikor jelenik meg a SuSE magyarított Office Desktop változata, aminek kipróbálását a SuSE lehetővé teszi.

Kosztadinovszki Norbert
(kosztadinovszki@diaelec.hu)