Végre tudtunk egy kis időt fordítani a weben is elérhető szótár (szotar.kiskapu.hu) frissítésére. Igaz, még mindig csecsemőkorát éli, de a tartalma folyamatosan bővül, és igyekszünk a jelzett hiányosságokat is javítani benne. Valójában nem is a szótár a legjobb meghatározás rá, hiszen egy szógyűjtemény ez, elsősorban informatikával foglalkozó fordítók számára, de természetesen bárki használhatja, aki kíváncsi, hogy melyik angol szakszót mire fordítjuk. Gondolom, mindenki számára magától értetődő, hogy egy-egy szónak rendkívül sok fordítása lehet, és nem csak amiatt, mert a szó különböző területeken különböző fogalmakat jelöl, de a fordító által használt stílus, környezet is befolyásolhatja. Egyfajta úttörő szerepet is vállalunk ezzel a szótárral, hiszen nemcsak a jó fordítások vannak benne, de gyakran a helytelen vagy egyenesen hibás fordítások is. Mivel valahogy jelölni szerettük volna, hogy mi melyik szót ajánljuk, a szavak mellett két jelzőt is használunk, az egyik a ***Besorolás***, a másik pedig az ***Ajánlás***. Így tehát a szótárban ellenjavallott szavakkal is találkozhatunk, amelyeket még egy kis halálfejes zászló is igyekszik megkülönböztetni a többitől (keressünk rá például a kliens szóra). A használatot tovább segítendő egyes szavakhoz a rendszer magától megjeleníti a hozzá tartozó címszavakat (ha például rákeresünk a sok vitát kiváltó ***mount*** szóra, a rendszer a ***befűz*** szócikket is megjeleníti), így is segítve a keresést. Ugyanígy igyekeztünk a magyarázatokat is hivatkozásokkal ellátni. Természetesen örök vitának számít nemcsak a mit fordítsunk le, hanem a hogyan is. Az általunk használt néhány irányelvet a nyitóoldal tartalmazza, bár a lista természetesen nem teljes. Ugyanakkor örömmel fogadok minden véleményt, ötletet, kérdést a szótárral kapcsolatban, különösen az új szócikkekre vonatkozó javaslatokat!

B/K – I/O

bővítmény (plugin) – Olyan modul, ami képes az adott programmal együttműködni és azt valamilyen további szolgáltatással kiegészíteni. Például a böngészőprogramok bővítményeket használnak a különböző (alapkiépítésben nem támogatott) fájltípusok kezeléséhez.

burkoló (wrapper) – Olyan program, ami a beérkező adatokat egy meghatározott protokoll szerint átvitelre készíti elő (ha kell, feldarabolja stb.). A burkoló által kapott adat(folyam)ot ezzel elrejti. Egy burkoló segítségével például egy csak TCP/IP-t engedélyező hálózati szakaszon keresztül is tudunk IPX-csomagokat továbbítani.

bus → sín

ciklus – lásd: loop

compatible – több értelemben használják. Amikor az az értelme, hogy a szóban forgó program egy másik program kiváltására, illetve a vele való együttműködésére alkalmas, akkor a csereszabatos fordítást használjuk, de előfordul, hogy az angol eredeti is pontatlan vagy olyan szavak helyett használja, mint például az „egyenértékű”.

csereszabatos – compatible

decoder → visszafejtő

dump → kiíratás

emacs – egy hosszú ideje használt, nagy tudású szerkesztőprogram (lásd az 52. oldalon lévő „Bevezetés az Emacs használatába” című cikket). Ami miatt a szótárba került, az a kiejtése körüli huzavona. A szó eredeti kiejtése ugyanis ***imeksz***, bár rendkívül sokan egyszerűen ***emacs***-nak ejtik. Ma már talán nem is az a fontos, hogy melyik kiejtést használjuk, hanem hogy mindkettőt megismerjük, ha halljuk.

erőforrás – resource

erő-visszacsatolás – lásd: force feedback

fejállomány – (header file) Programnyelveknél (például C, C++) azok az állományok, amelyeket több forrásfájlban is használni kívánnak. Ezek használatára az adott forrásban egyetlen parancssorral hivatkozhatunk (C esetén ez a `#include` parancs).

firewall – tűzfal

force feedback – elsősorban a játékkonzolok kapcsán megismert fogalom. Ezzel jelzik például az autóversenyhez használt kormányoknak azt a képességét, hogy a program által beállított mértékben „ellenáll”, vagy valamilyen erőhatást fejt ki. Nem létezik rá jó magyar fordítás, többen az erő-visszacsatolást vagy az erő-visszajelzést használják. Ötleteket várunk!

görgő – (wheel) az egereken található vezérlő, ami általában három gomb szolgáltatását egyesíti (fel, le és kattintás).

handler – kezelő

házirend – (policy) szabályok gyűjteménye

header – fejállomány

hurok – (loop) sok értelemben használják, ami miatt ismét a szótárba került, az az, hogy egyre gyakrabban találkozhatunk vele „ciklus” értelemben. Ez bár szakmailag szokatlan, nem helytelen (kétségtelenül rendhagyó, ha szóba kerül a főhurka).

I/O – B/K

jelzés – (signal) lásd a 48. oldalon lévő „Linuxos jelzések alkalmazásfejlesztői szemszögből” című cikket.

képpont – pixel

kezelő – handler

kiírat (dump) – ebben az értelemben nagy mennyiségű adatot „kiömleszt”, formázatlanul továbbít.

kompatibilis – csereszabatos, együttműködő. Lásd: compatible.

lemezrész – (partition) a lemeztípusú háttértárak egy egységként kezelhető területei.

loop – lásd: hurok

natív – A rendszer (vagy amihez viszonyítva értjük) „saját nyelvét beszélő”, kifejezetten a rendszerhez készült (fordított). Általában natív kódról beszélünk, ami azt jelenti, hogy a rendszer külön értelmező vagy fordító nélkül képes futtatni. Jó fordítást keresünk!

packet forwarding – csomagtovábbítás

partition → lemezrész

pixel – képpnot

plugin → bővítmény

policy → házirend

resource – erőforrás

rule – szabály

sequencer → sorrendvezérlő

set top box – Azokat az eszközöket nevezik így, amelyek valamilyen informatikai háttérrel támogatott médiaszolgáltatást végeznek otthoni felhasználásra, és egyszerűen a tévé vagy az erősítő „tetejére tehető dobozok”, például ilyenek internetkapcsolatot kezelni képes zenelejátszók, a merevlemezzel ellátott filmrögzítők. Fordítást keresünk!

signal → jelzés

sín (bus) – a gépen belül az elemek adatsínekkel vannak összekapcsolva, ezeken a síneken „közlekednek” az adatok.

sorrendvezérlő – (sequencer) a zeneprogramok világában használt szó, bővebben lásd a „Szintetizátorok Linuxon” című cikket a 28. oldalon. Pontos fordítást keresünk!

szabály – rule

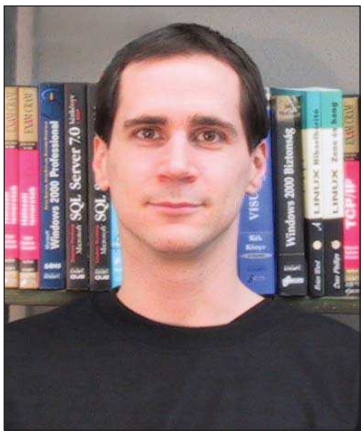
tűzfal – firewall

visszafejtő (decoder) – a sokszor használt „dekóder” szó magyar megfelelője.

wheel → görgő

wrapper → burkoló

Beköszöntő



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

Tombol a nyár, beköszöntött az AMD-tulajdonosok számára legszomorúbb időszak. Mert az odáig rendben van, hogy télen nem kell panaszkodni a központi fűtésre, de hogy a harminc-sok fokban is egy fűtőtestet kelljen dédelgetni az asztal alatt, az már sok. Nem baj, a lényeg, hogy az ember végre hódolhat kedvenc időtöltéseinek. Mert a linuxozók többsége kedvtelésből (vagy éppen feszültségcsökkentési célzattal) kezd el a pingvinekkel foglalkozni. És hogy az örök igazságra hivatkozzak: játszva tanul az ember. Ezzel bizonyíttatott is, hogy a játékoknak márpedig helye van. Bizony-bizony, a Linux komoly lemaradásban van (vagy talán csak volt?) a legújabb játékok terén. Szerencsére rendkívül sok játék linuxos átültetése készült már el, olyannyira, hogy egy külön rovatot is követelt magának. Így indítottuk útjára Játszótér című rovatunkat, amelynek keretében múlt havi számunkban kezdte meg cikksorozatát ***Kosztadinovszki Norbert***. A sorozat második részét a 76. oldalon találjuk.

És hogy a komolyabb oldalát is egy kicsit körbejárjuk, a 25. oldalon az SDL programkönyvtárról találunk egy cikket. Ez a könyvtár rejtőzik nagyon sok játék linuxos változata mögött (sőt nem csak Linux alatt), így aki játékfejlesztésre adja a fejét, annak mindenképpen érdemes megismerkednie vele.

De ne feledkezzünk meg azokról sem, akik a zene világában vannak otthon, hiszen a hangok világa is rengeteg embernek nyújt kellemes perceket. ***Dave Phillips***, a ***Linux Zene és Hang*** című könyv szerzője egy komolyabb lélegzetvételű cikkben (Szintetizátorok Linuxon, 28. oldal) vállalkozik egy csokorba fogni azokat a programokat, amelyeket ő maga is örömmel használ. Megjegyzem, nem véletlen, hogy a linuxos rendszergazdák többsége ódzkodik a zeneeszközöktől. Ugyanis – ahogy azt Dave is leírja – könnyen előfordulhat, hogy például egy nem teljesen támogatott, vagy hibásan felismert MIDI-eszköz miatt az egész rendszer dob egy hátast. Szerencsére a zeneszerzők általában nem a vállalat központi kiszolgálójára akarják kötni a keverőpultot. Erre sokan mondhatnátok, hogy egy kiszolgálóban általában nincs is hangrendszer, de láttam én már olyat, amikor a külön gépterembe berakott kiszolgálóban ott kuksolt egy AWE 32-es (akkoriban ez nagyon komoly kártya volt!).

Ha már szóba kerültek a rendszergazdák, egy érdekes cikkre szeretném felhívni minden rendszergazda (tehát a linuxosok döntő többségének) figyelmét. Az 56. oldalon ***Mick Bauer***, aki rendszeresen jelentet meg biztonsági témájú cikkeket a lap hasábjain, bemutatja a Firewall Builder program használatát. Pontosabban elkezdi a bemutatást, hiszen ez egy átfogó rendszer. Amikor először láttam a cikket, felrémlett bennem első találkozásom egy ilyen „grafikus felületű” tűzfalszabályokat kezelő programmal, a watchdoggal. A történet sajnos nem a boldog véggel fejeződött be, bár a program mentségére legyen mondva, hogy akkor még csak próbaváltozatban létezett. A másik oldalról nézve viszont tényleg nagy segítség egy rendszergazdának (főleg, ha „vizuális típus”), ha egy jól áttekinthető szerkezetben láthatja és kezelheti a beállításokat. Egy-egy komolyabb tűzfal kezelésénél (különösen, ha a szabályokat gyakran kell változtatni) egy ilyen program rendkívüli mértékben felgyorsíthatja a munkát.

Jó olvasást kívánok, remélem, mindenki talál a saját kedvéhez illő olvasmányt, és főleg, kényelmes nyugágyat kívánok a lap olvasásához, mondjuk, a Balaton partján!

Programvadászat

2.4.21-es rendszermag

Hosszas várakozás után megjelent a Linux-rendszermag 2.4.21-es megbízható változata. Jó néhány biztonsági rést betömtek a fejlesztők, a kód szinte egyetlen része sem maradt érintetlen a tavaly novemberi 2.4.20-as kiadáshoz képest. Fontos figyelmeztetést olvashatunk több linuxos weboldalon arra vonatkozóan, hogy mindenképpen valamelyik hivatalos rendszermagtükörről szedjük le a forrást (ahogyan ezt mi is tettük a korongunkra felkerült csomag esetében), mivel az utóbbi évben többször is előfordult, hogy bizonyos programok (Sendmail, `tcpdump`) forráskódja a fordítás során aktiválódó, a fordítást végző felhasználó jogosultságaival futó trójai programot tartalmazott. A rendszermag esetében GPG-vel tudjuk ellenőrizni a csomaghoz tartozó aláírást. Az ehhez tartozó útmutatást a ➲ http://www.kernel.org/signature.html oldalon találjuk. Korongunkon a kísérletező, esetleg a 2.5-ös rendszermagot jobban megismerni vágyó olvasóink kedvéért a legújabb fejlesztői rendszermag is megtalálható.

Samba 3.0 Beta

A Samba-csapat bejelentette a Samba 3.0.0 kódjának első próbaváltozatát. Bár ezzel a lépéssel a végleges 3.0-s változat már elérhető közelségbe került, a fejlesztők arra figyelmeztetnek, hogy éles rendszeren még ne próbálja ki senki!
Az újdonságok közül álljon itt kettő:

- Active Directory-támogatás,
- Unicode-támogatás.

PostgreSQL 7.3.3

Az általam igen kedvelt nyílt forrású adatbázis-kezelő, a PostgreSQL adatbázis legújabb változata a 7.3.3-as. A változások teljes listája a ➲ http://www.us.postgresql.org/postgresql-7.3.3/release.html#RELEASE-7-3-3 címen érhető el.

Mozilla 1.4RC1

A következő megbízható Mozilla az 1.4-es sorozatból is kikerülhet, mint azt a korongunkon megtalálható RC1-es jelölésű fájl is mutatja. Sok újdonság lesz ebben a kiadásban az előzőekhez képest. Próbálgatni, ízlelgetni érdemes, viszont vigyáznunk kell vele, mert hibás működésre hajlamos!
A telepítés lépései a következők:

1. Fontos, hogy azzal a felhasználóval, amelyikkel a rendszerünkre telepítettük, egyszer mindenképpen futtassuk a böngészőt, hogy az „autoregistration” folyamat zavartalanul lefuthasson és a ***compreg.dat*** fájl létrejöjjön a telepítési könyvtárban. Java appletek futtatásához a ➲ http://home.netscape.com/plugins/jvm.html címről telepíteni kell a JRE v1.3 bővítményt.
2. A korongon lévő fájlt kicsomagolva egy ***Mozilla-installer*** könyvtárat kapunk, lépjünk be ebbe a könyvtárba, és a `./Mozilla-installer` paranccsal futtassuk a telepítőprogramot. A képernyőn megjelenő utasításokat végigkövetve a telepítővarázsló végigvezet bennünket a telepítés egészén – csak néhány egyszerű kérdésre kell válaszolni, például hogy melyik könyvtárban szeretnénk telepíteni a programot. A Mozilla indításához lépjünk be a telepítési könyvtárba, és adjuk ki a `./Mozilla` parancsot. Kellemes böngészést!

Játékok

Játékrovatunkhoz ismét bőségesen akad telepíteni-, nézegetnivaló anyag, a kedves olvasók ezt szokás szerint a CD ***Jatek*** könyvtárában találhatják meg.

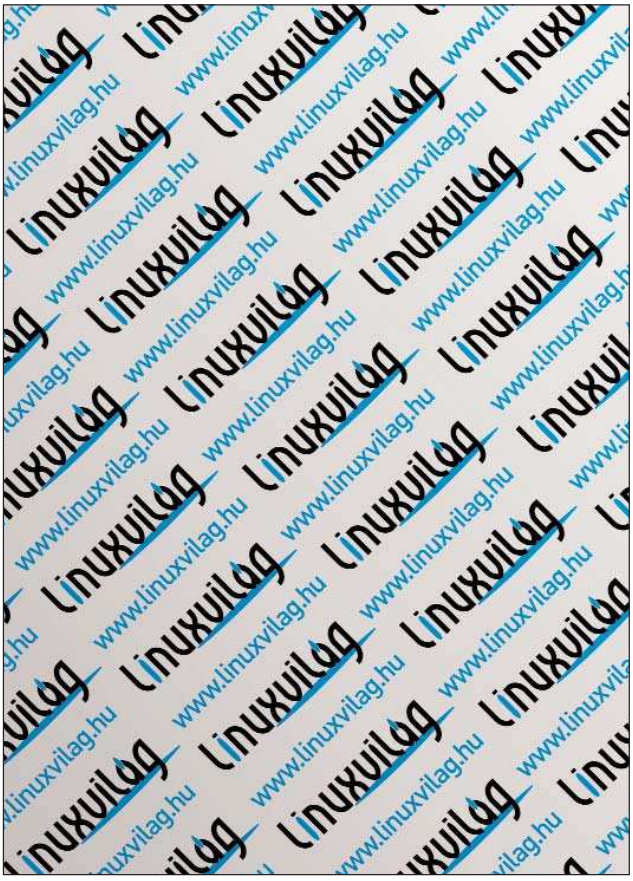
Magazin

Természetesen a magazin cikkeihez tartozó program-, forráskód-összeállítások és bővítmények sem hiányozhatnak a korongról.

Szótár

A ➲ szotar.kiskapu.hu teljes anyaga is megtalálható a korongunkon, sajnos a keresőrésze még nem működik, a szavakat ábécérendben érhetjük el.

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



Kártyaolvasó beágyazott Linuxszal

A Sharp – RK40PR004 jelzéssel – olyan fizikai kapcsolatot nem igénylő IC-kártyaolvasó készüléket jelentett be, ami beágyazott operációs rendszerként Linuxot futtat. A készülék alkalmas a szabványos intelligens kártyák kezelésére – ilyeneket egyébként leginkább irodai beléptető rendszereknél vagy díjszámító alkalmazásokban láthatunk. A Sharp új leolvasójához nincs szükség terminálra vagy számítógépre, ami a vezérlést elvégezné, ezeket a feladatokat saját operációs rendszere látja el. Mivel ennek köszönhetően külön hálózatot sem kell kiépíteni, lényegesen leegyszerűsödik és olcsóbbá válik a telepítés, a beágyazott Linux révén pedig saját kártyaolvasó alkalmazások fejlesztése is lehetővé válik.

➲ http://www.sharpsma.com

Négy színcsatornás Samsung TFT

A Samsung bemutatta a világ első olyan TFT-LCD-it, amelyek a megszokott három helyett négy színcsatornát kezelnek. A jelenlegi kijelzők három csatornával működnek (vörös, zöld, kék), a Samsung ezeket egy fehér színcsatornával egészítette ki. A fehér színt a mostani kijelzők úgy állítják elő, hogy a háttérvilágítás fényét átengedik az RGB-maszkon – ez azonban közel egyharmaddal csökkenti a fényerőt. A fehér alképpont hozzáadásával ez a jelenség megszűnik, a háttérvilágítás fehér fénye akadálytalanul átjuthat a maszkon, így a kijelző fényereje 30–70 százalékkal nagyobb lehet, miközben az energiaigénye változatlan marad. A negyedik alképpont beépítésének ötlete nem volt új, ám eddig nem tudták úgy megoldani, hogy a fényerő ne csökkenjen, ne növekedjenek a gyártási költségek, illetve ne torzuljanak a színek; a Samsung ezeket a hibákat újfajta képponttömbökkel és új színfeldolgozó eljárásokkal hárította el.

➲ http://www.samsungsemi.com

Játékos Sun

A Sun Microsystems a jövőben a játékok piacáról is szeretné kihasítani a maga részét. Ennek jegyében a cég létrehozta a Game Technologies Groupot, amelynek feladata a sokmillió dolláros játékpiac megdolgozása lesz. A Sun természetesen nem akarja megváltani a világot, hanem a Java és a Solaris terjedését szeretné elérni a hálózati játékosok és fejlesztők körében, legyen szó akár asztali számítógépekről, akár konzolokról vagy – a játékpiacon egyre fontosabbá váló – mobil készülékekről. A Java alapú játékok szinte bármilyen géptípuson futtathatók, így a fejlesztők egyetlen termékkel is hatalmas piacot érhetnek el, de a Java – és vele a hálózati megoldások terén elismert Sun – a kiszolgálóoldalon, a szolgáltatások fejlesztésekor is megtalálhatja a maga szerepét. A szolgáltatóknak nem csupán egyszerű játékokat kell elérhetővé tenniük, hanem több személy által egymás elleni játéklehetőséget kínáló, akár több ezer felhasználó számára is egyidejű szolgáltatást kínáló környezetet kell létrehozniuk, aminek ráadásul nagyon sokféle felhasználóoldali készüléket kell támogatnia.

➲ http://community.java.net/games/

Kézikönyv a biztonságiaknak

Elkészült a CERT Coordination Center által még 1998 decemberében megjelentetett „Handbook for Computer Security Incident Response Teams (CSIRTs)" című összeállítás második kiadása. A dokumentumot olyan meglévő vagy újonnan formálódó munkacsoportoknak ajánlják, amelyek elsődleges feladata a számítógépes biztonsági kérdések megoldása, a kialakult vészhelyzetek kezelése. A dokumentum a munkacsoport működésétől, a belső adatáramlástól kezdve a házirendeken keresztül egészen a minőségbiztosításig számos témakört érint, így valószínűleg nem csak a biztonsággal foglalkozó szakemberek számára jelenthet érdekes olvasmányt.

A dokumentum pdf formátumban a következő címről tölthető le:

➲ http://www.cert.org/archive/pdf/csirt-handbook.pdf

Szórakozásra kihegyezve

Szokatlan hordozható gépet mutatott be a Toshiba. A Satellite 5205-S705 jelzésű gépre a Microsoft Windows XP Media Center Edition változata került, ugyanis a Toshiba igazi hordozható multimédiás

szórakoztató eszközt akart varázsolni belőle, ami a munka mellett tévézésre, az adások felvételére, DVD-lejátszásra, vezeték nélküli hálózati kapcsolaton keresztüli internetezésre, CD-írásra egyaránt alkalmas. A gépbe 2,4 GHz-es Intel Pentium 4 processzor, 512 MB memória és 60 GB-os merevlemez kerül, az nVidia GeForce FX Go 5600 VGA-vezérlő egy 15"-os kijelzőt vezérel, multimédiássá pedig a beépített tévévevő, a távirányító és a beépített, mélynyomóval kiegészített Harman/Kardon hangszórók teszik. Ára 2700 dollár, vagyis nagyjából 600 000 forint.

ATI – MontaVista partnerség

A MontaVista a továbbiakban támogatni fogja az ATI XILLEON nevű, a teljes rendszert egyetlen lapkán tartalmazó termékét. A XILLEON egy 300 MHz-es MIPS processzort, kettős grafikus vezérlőt, hangvezérlőt, MPEG-2 dekódoló egységet, illetve olyan ki- és bemeneti – köztük PCI- és USB-kapcsolat, illetve merevlemez kezelésére alkalmas – vezérlőket tartalmaz, amelyekre a set top boxokban vagy digitális tévékben lehet szükség. A MontaVista Linux Professional Edition ATI XILLEON lapkákra igazított változata nyár közepétől lesz elérhető.

Elkészültek a 2003-as SOT-termékek

A SOT bejelentette a SOT Linux és a SOT Office 2003-as változatát. Az asztali és kiszolgálóváltozatban egyaránt elérhető operációs rendszer, illetve az irodai csomag bárki számára ingyenesen letölthető. A SOT korábbi változataiban viszonylag sok egyedi fejlesztésű összetevőt lehetett találni, az új változatokban azonban ezek nagy részét szabad programok váltották fel, így az egész terjesztés szabadon hozzáférhető, módosítható, beépíthető és továbbadható. Az új terjesztés megfelel a Linux Standard Base 1.2 előírásainak, összeállításakor a biztonságra, a termelési környezetben való azonnali alkalmazhatóságra helyezték a hangsúlyt.

➲ http://www.sot.com

Színes tintákkal álmodva

A Xerox kutatói olyan különleges nyomtatási megoldást mutattak be, ami lehetővé teszi, hogy egymás fölé több képet is nyomtassunk, majd a hordozót a megfelelő fénnyel megvilágítva ezeknek csak az egyikét jelenítsük meg. Ez az első alkalom, hogy jó minőségben sikerült ilyen, a fényforrás váltásával megjeleníthető–eltüntethető képeket előállítani. A „Switch-A-View" megoldás a reklámszakmától kezdve egészen a biztonsági vagy postai szolgáltatásokig rengeteg területen alkalmazható: például pólókra vagy különféle ételek csomagolására rejtett képeket lehet nyomtatni, amelyek közül eltérő fényviszonyok között mindig mások tűnhetnek fel, illetve a fényforrás változtatásával érdekes hatások hozhatók létre, akár egyszerű mozgóképek is lepörgethetők.

➲ http://www.xerox.com/innovation

Pontosabb szűrés az MX Logic keretrendszerével

Az MX Logic új megoldással bővítette Spam Guard szolgáltatását. Az MX Logic Stacked Claffication Framework nevű keretrendszerének lényege, hogy több szemétszűrő együttműködve válogatja ki azokat a leveleket, amelyek nagy valószínűséggel kéretlenül érkeztek a címzetthez. Ez a keretrendszer bővült most a Distributed Checksum Clearinghouse (DCC) összetevővel, ami – a víruskeresőknél alkalmazotthoz hasonló módszerrel – minden levélről egy ellenőrzőösszeget készít, majd ezt egy adatbázisban helyezi el. Ha az adatbázisban túlságosan sokszor szerepel egy ujjlenyomat, akkor az nagy valószínűséggel valamilyen körlevél, így érdemes megfontolni a kiszűrését. A keretrendszer azonban ennél többre is képes, segítségével több szemétszűrő – fekete- és fehérlisták, heurisztikus elemzés és Bayes-döntés alapú – is megvizsgálhatja a leveleket, majd a szavazataik alapján dől el, hogy az adott levél kéretlen-e vagy sem. Ezzel a módszerrel – a cég állítása szerint – 98,5 százalékos hatékonysággal kiszűrhetők a felesleges levelek, és a rendszer nagyon kevés üzenetet nyilvánít alaptalanul kéretlennek.

➲ http://www.mxlogic.com

Újabb Zaurusok a Sharptól

Két új Zaurus modellt mutatott be a Sharp SL-C760 és SL-C750 típusjelzéssel. Az új gépekbe a korábbinál erősebb Intel Xscale processzor és kétszer több – 64 MB – memória került, 3,7"-os érintőképernyőjük 640×480 képpont és 65 000 szín megjelenítésére képes. A Zaurusok az előtelepített programoknak és az alapgép tudásának köszönhetően eleve sokoldalúak, és mikrofon, digitális kamera vagy éppen külső VGA-kártya csatlakoztatásával szolgáltatásaik köre tovább bővíthető, üzleti és személyes célokra egyaránt alkalmassá téve őket. Sajnos a Sharpot alighanem elkényeztetik a japán vásárlók, mivel hivatalos támogatást és adatokat inkább csak japánul találunk, nekünk csak az a boldogító tudat jut, hogy valahol a világban linuxos zsebtitkár is létezik.

Münchenben a nyílt forrás nyert

München vezetősége úgy döntött, hogy a város 16 ezer felhasználójának 14 ezer személyi számítógépét a jövőben Windows helyett Linux operációs rendszerrel fogják használni. A képviselők gondosan megvizsgálták, hogy a jelenlegi Windows NT és Microsoft Office összeállítás helyett továbbra is a Microsoft termékeit – Windows XP és Microsoft Office – használják, vagy Linuxra és OpenOfficera váltsanak. Hosszabb távra szóló döntésükben elsősorban az játszott közre, hogy a nyílt alkalmazások révén szállítótól, gyártótól jelentős részben független, és így – bizonyos szempontból – biztonságosabb rendszert építhetnek ki. A város jövő év tavaszáig dolgozza ki az áttérés menetrendjét, az értékesítési oldal főszereplője várhatóan a SuSE AG és az IBM Germany lesz.

Kedvez a felsőoktatásnak a Zend

Az izraeli Zend Technologies bejelentette, hogy Zend Studio nevű termékét ingyenesen érhetik el azok a felsőoktatási intézmények, amelyek a PHP használatát órarendi keretben oktatják. A Zend emellett az ilyen iskoláknak összes termékére 25 százalékos engedményt kínál. A PHP az egész világon viharos gyorsasággal terjedt el az egyetemeken és főiskolákon, sok helyen nemcsak tanítják használatukat, de az egyetemi portált vagy webes levelezőrendszert is PHP segítségével működtetik. Mivel könnyen tanulható, gyorsan megszerethető, az általános parancsfájlkészítés is hamar belopható általa a diákok szívébe. A közleményből nem derül ki, hogy a Zend a világ összes egyetemére kiterjeszti-e a kedvezményt, vagy csak az amerikaiakra.

➲ http://www.zend.com

Unixot vásárol a Microsoft

A SCO Group – mint a Unix operációs rendszer tulajdonosa – bejelentette, hogy a Microsoft Corporation megvásárolta tőle a Unix operációs rendszernek és forráskódjának használati jogát, csatlakozva ezáltal ahhoz a többezer vállalathoz és szervezethez, amelyek rendelkeznek ezekkel a jogokkal. A cég képviselője szerint a Microsoft ezzel is kimutatja, hogy tiszteletben tartja a szellemi alkotásokhoz, programokhoz fűződő tulajdonjogokat; illetve a mostani megállapodás is hozzájárul ahhoz, hogy a Microsoft termékei teljes mértékben képesek legyenek együttműködni a Unix alapú megoldásokkal. Az, hogy a Microsoftnak pontosabban milyen szándékai vannak a forráskóddal, a cég közleményéből nem derül ki.

➲ http://www.sco.com

Memóriakártyával bővíthető flashmeghajtó

A Kanguru Solutions Micro CF néven olyan USB-s flashmeghajtót jelentett be, ami a CompacFlash kártyák és az IBM/Hitachi MicroDrive-meghajtók kezelésére is képes. A flashmeghajtók a memóriakártyákhoz hasonlóan rohamosan terjednek, ám sokszor gondot jelent, hogy nincs kéznél megfelelő kártyaolvasó. Ha ilyen meghajtónk lesz, ez az apróság is ki lesz pipálva, ha pedig a meghajtó 32, 64 vagy 128 MB-nyi tárhelye kevésnek bizonyulna, akkor csak egy marékra való memóriakártyát kell magunkkal vinnünk. Ha a flashmeghajtót memóriakártyával is bővítjük, akkor a rendszerben két meghajtó jelenik meg, ezeket külön használhatjuk. A mindössze 20 grammos apróság ára 40 dollártól indul, egyetlen hibája, hogy csak USB 1.1-es felülettel rendelkezik. A cégnek van már hasonló terméke, ez a KanguruMicro Drive+ a Secure Digital és a Multimedia Card kártyák kezelésére képes.

Hasonlóan érdekes a Kanguru másik újdonsága, az első 2 GB kapacitású flashmeghajtó, ami szerencsére már USB 2.0 felülettel bír. A meghajtó két különleges könyvtárat tartalmaz, az egyik ***Zip***, a másik ***Secret*** névre hallgat – ha valamilyen anyagot tömöríteni vagy titkosítani szeretnénk, ezekbe kell másolnunk őket. A titkosító–tömörítő eszköz ára ugyancsak 40 dollártól kezdődik.

➲ http://www.kanguru.com

A Palm felvásárolja a Handspringet

A Palm és a Handspring igazgatótanácsa bejelentette, hogy a Palm felvásárolja a Handspringet, egy erősebb, a Palm OS alapú intelligens telefonok és zsebtitkárok piacán vezető céget hozva létre ezáltal. Mindeközben a programfejlesztéssel foglalkozó PalmSource véglegesen elszakad a Palmtól. A zsebtitkárok területén jelentős, várhatóan a kínálatra és a piaci erőviszonyokra is fontos hatást gyakorló felvásárlás befejezése őszre várható. Az egyesülés után létrejövő vállalat valószínűleg még ebben az évben nevet is fog változtatni. A Handspring és a Palm története kisebb kalandregény, amelyben a szétváló szálak meglepő fordulattal fonódnak újra egybe. A Palmot ***Jeff Hawkins*** és ***Donna Dubinsky*** alapította 1992-ben. A céget 1995-ben vásárolta fel az U. S. Robotics, amire viszont a 3Com tette rá a kezét két évvel később. 1996-ban jelent meg az azóta történelemmé lett PalmPilot 1000 és 5000. Hawkins és Dubinsky 1998-ban elhagyta gyermekét, és létrehozza a Handspringet, ami ugyancsak Palm OS alapú eszközöket kezdett forgalmazni. A Palm hamarosan hűtlen lett a 3Comhoz, egy részvénykibocsátás révén a vállalat önállósult, majd lassacskán megformálódott az az elképzelés, hogy az eszközöket és az operációs rendszert két külön vállalat fejlessze – így alakult ki a Palm Solutions és a PalmSource, az utóbbiba a Sony is befektetett. A történet napjainkban ér véget, amikor a Handspring – és megfelelő tisztségeket kapva annak alapítói is – visszatérnek a Palmhoz.

➲ http://www.palm.com
➲ http://www.handspring.com

MySQL AB-SAP-együttműködés

A MySQL AB bejelentette, hogy együttműködésbe kezdett a világ legnagyobb üzletialkalmazás-fejlesztőjével, az SAP AG-vel. A MySQL és az ugyancsak nyílt SAP DB adatbázis fejlesztői a továbbiakban közösen kínálnak adatbázis-rendszereket a kis- és közepes méretű vállalkozások számára, illetve a jövőben ezek fejlesztésében is együttműködnek. A MySQL AB a továbbiakban három terméket kínál majd: a MySQL Classic elsősorban a webhelyek üzemeltetői számára, naplózásra és beágyazott alkalmazásokhoz készül, a MySQL Pro nagyteljesítményű tranzakciókezelésre is képes lesz, míg a harmadik termék egy átkeresztelt SAP DB lesz, amelyet nagyvállalati adatbázis-kezelő rendszerekbe szánnak. A fejlesztők nagy hangsúlyt fektetnek a kétfajta adatbázisrendszer közötti együttműködésre is, ami a felhasználóknak az áttérést és az alkalmazásfejlesztést egyaránt megkönnyíti.

➲ http://www.mysql.com
➲ http://www.sap.com

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

Linux-index

1. Az informatikai kiadások várható növekedési aránya 2003-ban: **4%**
2. A kiszolgálók számának várható növekedési aránya a vállalati kiszolgálók körében: a 2002-ben tapasztalt több mint ötven százalékos növekedésen túl további **40 százalékos** növekedés várható a Linux-kiszolgálók körében 2003-ban.
3. A Linux várható növekedési aránya Ázsiában 2003-ban: **24%**
4. A Microsoft Windows várható növekedési aránya Ázsiában 2003-ban: **6%**
5. Az IBM Linuxból származó bevétele az IBM saját bevallása szerint (milliárd dollár): **1**
6. A HP Linuxból származó bevétele a HP saját bevallása szerint (milliárd dollár): **2**
7. Ennyi milliárd dolláros kártérítést követel az SCO az IBM-től az SCO-engedéllyel rendelkező AIX forráskód révén kiszivárogtatott üzleti titkokért: **1**
8. Ekkora költségmegtakarítási arányt tapasztalt a Merrill Lynch brókerház, amióta az IBM nagygépeken Linuxot futtat: **40–50%**.
9. Legalább ennyi millió dolláros éves megtakarítást ért el a Merrill Lynch azáltal, hogy teljes körben alkalmazta a linuxos nagygépstratégiát: **100**
10. Ennyi linuxos gépet használnak jelenleg a Morgan Stanleynél: **400**
11. Ennyi további linuxos gép bevetését tervezi a Morgan Stanley: **300**
12. A Morgan Stanley ekkora ár/teljesítmény növekedési szorzót tapasztalt a hat négyprocesszoros Linux-gépen: **13**
13. Ennyi százalékos költségjavulást ért el a Lehman Brothers a Linux jóvoltából: **50**
14. Ennyi százalékos teljesítményjavulást ért el a Lehman Brothers a Linux jóvoltából: **20**
15. A háztartásokba bekötött normál telefonkábel-sávszélesség teljesítőképességének jelenlegi kihasználtsága: **1%**
16. A jelenleg eladott IBM-kiszolgálók ennyi százalékán fut Linux: **15–20**
17. A Linux-felhasználók számának éves növekedési aránya az elkövetkezendő pár évben *Scott McNealy*, a Sun ügyvezető igazgatója szerint: **30%**

Források
1–2.: Aberdeen Group
3–4.: International Data Corp., az Economic Times magazinban megjelent cikk alapján
5–6.: eWeek
7.: SCO
8–14.: Risk Waters Group
15.: *Bob Frankston*
16.: *Jim Stallings*, IBM
17.: Associated Press

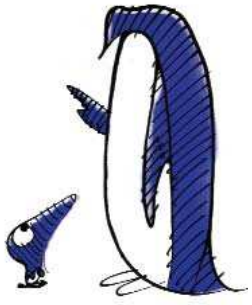
Linux Journal 2003. június, 110. szám

Rendszermag-fejlesztési hírek

Megszületett a FUSE Project (fájlrendszer a felhasználói területen) 1.0-s változata. A Unix egyik alapötlete az, hogy olyan eszközöket kell adni, amelyek általános, együttműködő módon végzik el az alapvető műveleteket. Ezek az eszközök általában a fájlrendszerben fájlként tárolt adatfolyamokkal dolgoznak. Egyes eszközök, például az SSH, az FTP és a tömörítőprogramok azonban kilógnak a sorból, mivel a fájlokat nehezen kezelhető formátumúra alakítják, vagy a hálózaton elérhető más rendszerekkel működnek együtt. Ismét más eszközök ragaszkodnak a saját belső vezérléshez, így egyetlen alapvető Unix-eszköz használatát sem teszik lehetővé. A FUSE segítségével a felhasználó az ilyen programok bármelyikét egy fájlrendszerszerű felület mögé rejtheti, s így minden művelet fájlműveletként végezhető, kihasználva az alapvető unixos eszközök gazdag kínálatát.
Elkészült a 2.4-es változatban megjelent virtuális memória (VM) alrendszer részletes leírása – köszönet érte *Mel Gorman*-nak. Hajdanán *Linus Torvalds* a teljesen új VM alrendszert belepottyantotta a Linux-rendszermagjába, egyenesen a megbízható kiadás kellős közepébe. Ez igen nagy port vert fel a fejlesztők körében, részben amiatt, hogy *Andrea Arcangeli*, az új kód szerzője gyakorlatilag semmilyen leírást nem mellékelt hozzá. Melnek hathavi munkájába került – jóval többe, mint eredetileg gondolta volna –, mire elkészítette a teljes VM működésének átfogó magyarázatát, illetve magát a forráskódot megjegyzésekkel látta el.
A rendszermag hibakövetési adatbázisához (Kernel Bug Database) részletes leírást készített az adatbázist létrehozó *John Bradford*. Ez az adatbázis az éppen időszerű Bugzilla-adatbázis hibáinak kiküszöbölésére jött létre, és számos fejlesztő használja. A hibakövetési adatbázis nem a szokványos módon működik, és a Linux-rendszermag sajátos igényeihez igazodó szolgáltatásokat kínál, például a *.config* fájl beállításai szerinti keresést.
A dinamikus rendszermagmodul támogatása (DKMS) a Dell egyik fejlesztőcsapatának műve. A GPLed-fejlesztés célja, hogy lehetővé tegye, hogy az eszközvezérlő-programok forráskódja a fájlrendszerben bárhol lehessen, ne csak a rendszermag forrásfájában. 2003 márciusában ott tartunk, hogy a DKMS a 2.4-es változatra jellemző, és nem veszi figyelembe a 2.5-ös változatban megjelenő jelentős átdolgozásokat, amelyek különösen a modul kódját érintik.
Több csoport is dolgozik az IPv6 szabvány IPSec protokolljainak megvalósításán. Az IPSec protokollkészlet olyan keretrendszer, ami támogatja a titkosítást és a hitelesítést az IP-címek szintjén, míg az IPv6 célja a rendelkezésre álló IP-címek számának a növelése. Habár az IPv6 széles körben még nem terjedt el, fontos a háttér folyamatos fejlesztése, hogy egy napon majd át lehessen rá térni a betegeskedő IPv4 szabványról. *Kazunori Miyazawa*, *Kunihiro Ishiguro*, *Hideaki Yoshifuji* és *Mitsuru Kanda* egyesült erővel azon munkálkodik, hogy az IPv6 IPSec támogatása működjön a 2.5-ös rendszermagfában.

Zack Brown

Linux Journal 2003. június, 110. szám



A Linux Journal honlapján számtalan gond megoldásához találhattok további segítséget. A Sunsite tükröoldalait, 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

IP-cím beállítása SuSE környezetben

SuSE 8.0-t üzemeltetek IBM ThinkPad 600E gépemen, és a munkahelyi kiszolgálót szeretném elérni. Hogyan állítsam be a TCP/IP-t, hogy magától felismerje a címeket?
Layla, satchumwatch@netscape.net

Futtasd a YaST2-t, a SuSE beállítóeszközét, és lépj a *Hálózati cím beállítása* képernyőre. Válaszd az *Automatikus cím beállítása (DHCP)* lehetőséget, ha van a hálózaton DHCP-kiszolgáló; vagy válaszd a *Statikus cím beállítása* lehetőséget, és írd be kézzel az IP-címet és az alhálózati maszkot.
Don Marti dmarti@ssc.com

Nem sikerül elindítanom az új telepítést

Amikor megpróbálom elindítani a Red Hat 7.3-at, érvénytelen rendszerlemezre utaló hibaüzenetet kapok. Azt is írja az üzenet, hogy cseréljem ki a lemezt, és üssek le egy billentyűt. Hogyan szüntethetném meg ezt a hibát a Linux újratelepítésén kívül?
Logan, crossl@lakecitycc.edu

Ez az üzenet általában azt jelenti, hogy a BIOS nem talált rendszerbetöltőt a lemezen. Ha a Linux telepítésekor telepítettél LILO-t vagy egy másik rendszerbetöltő programot, akkor valószínűleg ez a telepítés nem volt sikeres, és ellenőrizned kell a használt jellemzőket. Ha nem telepítettél rendszerbetöltő programot, a helyzet egyszerűbb. Akármi is történt, a vészindító lemez vagy az eredeti telepítőlemez segítségével elindíthatod a rendszert. Ezután telepítsd újra a rendszerbetöltőt.
Chad Robinson, crobinson@rfgonline.com

Ha a hajlékonylemezes meghajtóban benne maradt egy lemez, akkor távolítsd el.
Christopher Wingert, cwingert@qualcomm.com

PowerPC alaplap?

A bátyám és én linuxos PowerPC számítógépet szeretnénk építeni. Van ötletetek arra nézvést, hogy hol kapható PPC-hez való alaplap? A PPC-s Linux-terjesztések közül ismerjük a Yellow Dog, a Mandrake, a SuSE, a Rock, a Holon, a Debian, a Vine és a Gentoo változatokat. Ezek szerint van egy pár Linux-terjesztés a PPC-hez.
Rick Killingsworth, iamrlk@yahoo.com

A következő hivatkozások talán érdekesek lesznek; PPC alapú számítógépekről és a rajtuk futó Linuxokról szólnak: lppcfom.sourceforge.net és linuxppc64.org. Az utóbbi hivatkozás az IBM 64 bites PPC vasáról szól. A ➲ http://www.openppc.org a nyílt PPC-ről tartalmaz érdekességeket, és néhány alaplapról diagram és terv is megtalálható rajta.
Felipe E. Barousse Boué, fbarousse@piensa.com

Öt év hiba nélkül – mostanáig

SuSE 6.0-s rendszerem gyakorlatilag szünet nélkül kitűnően működött az elmúlt öt évben – kivéve az áramszüneteket. Névkiszolgálót és Sendmailt futtattam rajta. Egyszer csak nem tudtam bejelentkezni abba a felhasználói fiókba, amit mindig is használtam. Csak a rendszergazda és egy másik felhasználói fiók tulajdonosa tud bejelentkezni, de nem tudok rájönni, hogy miért pont az a bizonyos fiók viselkedik ilyen különösen. Bár bejelentkezni nem lehet, az `su` paranccsal és a helyes jelszó megadásával bármelyik fiókra át tudok váltani. A fiókok nincsenek zárolva, a jelszavak nem jártak le és helyesek, a felhasználóknak jogosultságuk van a saját könyvtárukra, a *passwd* és a *shadow* állományok hozzáférési jogosultságai rendben vannak. Próbáltam egy fiókot ugyanazzal a csoportazonosítóval (admin) és csoporttal létrehozni, mint a működő fiók, de nem működött. A rendszernaplóba a helytelen jelszó üzenet íródik.
Juan Alvarez, juan.alvarez@thales-is.com

Anélkül, hogy jobban megvizsgálnánk a rendszeredet, ez a helyzet arra utal, hogy a gépedet feltörték. A `telnet` egyszerű szöveges jelszavakat küld át a hálózaton, és egy ötéves terjesztésben a telepített démonoknak számos sebezhetőségi pontja ismert. Nehézségeid erre utalnak. Ha ezt a lehetőséget kizárhatod, akkor vizsgáld meg a felhasználók bejelentkezését befolyásoló egyéb szolgáltatásokat. Például van-e */etc/nologin* állomány? Ez megakadályozza a nem rendszergazdai felhasználók bejelentkezését, és különleges felhasználói fiókod talán azért élvez más elbánást, mert tagja a rendszergazdai csoportnak a */etc/group* állományban. Vizsgáld meg a */etc/passwd* állományt is, hogy minden felhasználónak van-e érvényes parancsértelmező héja és saját könyvtára.
Chad Robinson, crobinson@rfgonline.com

A telepítés korára való tekintettel megfontolhatnál egy újabb és biztonságosabb terjesztésre történő frissítést. A második tippem az lenne, hogy a lemez betelt.
Christopher Wingert, cwingert@qualcomm.com

Két jó tanács a biztonsági gondok megelőzésére:
1. távolítsd el vagy tiltsd le a nem használt programokat, de a `telnet`-et mindenképp – használj OpenSSH-t,
2. iratkozz fel a terjesztésed biztonsággal foglalkozó levelezési listájára, hogy értesülj a frissítésekről, és azonnal telepítsd a frissítéseket.
Don Marti, dmarti@ssc.com

Az SMP-rendszer nem kapcsol ki

Red Hat 8-at futtatok SMP-vel. Megfigyeltem, hogy amikor egyprocesszoros rendszermaggal dolgozom, a számítógép rendesen kikapcsolódik. Ha rendszerindításkor az SMP-rendszermagot választom, és utána állítom le a gépet, a rendszer leállítja az összes folyamatot, majd kiírja, hogy most már kikapcsolhatom a számítógépet. Az a kérdésem, hogy miért nem kapcsolódik ki a számítógép, ha az SMP-t használom.
Ron Oliva, rmoliva@citlink.net

A kikapcsolás Linuxon egy APM-hívást használ, de az APM SMP-módban nem biztonságos, ezért a Linux

letiltja. Van egy olyan, a rendszermagnak kiadható parancs, ami lehetővé teszi, hogy a rendszermag csak annyit engedélyezzen az APM szolgáltatásaiból, amennyi a kikapcsoláshoz kell. Az újabb rendszermagoknál hozzáadhatod ezt az `append=` `sort`-ot a `lilo` vagy a `grub` beállítóállományához:

```
apm=power-off
```

Marc Merlin, marc_bts@google.com

Nincs inetd, de telnetelni kellene

Az `inetd` gyakran leáll a rendszeremen (Slackware 2.0.28), emiatt a gép `telnet`-tel nem érhető el. Van-e könnyű módja az `inted` önműködő újraindításának, ha meghal?
Mark Johnson, Mark.Johnson@InfoHarvest.ca

Az általad futtatott `inetd`-változat ősrégi, és bizonyos változatokkal voltak megbízhatósági gondok. Sokféle megoldás létezik: frissíthetnéd a rendszeredet; az újabb változatok általában jól működnek, és egyéb előnyös tulajdonságokkal is rendelkeznek. Ha az `inetd` lecserélésén gondolkodsz, próbáld ki az `xinetd` vagy a `daemontools` programot, mindkettő eléggé megbízható és sokoldalú. Ha a `cron` szolgáltatás megbízhatóan működik, és nem szükséges az `inetd`-t a halála után azonnal újraindítani, körülbelül ötpercenként megpróbálhatod futtatni a következő parancsfájlt a `cron`-ból:

```
#!/bin/sh
ISINETS=`ps ax | grep inetd | \
grep -v grep | wc -l`
  if [ $ISINETD != 1 ]; then
     /usr/sbin/inetd
  fi
```

Chad Robinson, crobinson@rfgonline.com

Cseréld le a `telnet`-et az OpenSSH-ra, máskülönben a jelszavaid és egyéb érzékeny adataid ellophatók a hálózatról. Az OpenSSH ügyfél és kiszolgáló minden Linux-terjesztésben megtalálható, és a kiszolgálóhoz való ügyfelek minden elterjedt felületen léteznek. Az SSH-t ugyanolyan könnyű használni, mint a `telnet`-et, de az SSH a kapcsolatot titkosítja.
Don Marti, dmarti@ssc.com

A lemezfelosztási táblázat nem változik

Seagate ST32550 SCSI merevlemezt használok AHA1720 illesztőkártyával, de az `fdisk` programmal nem tudom a lemezt felosztani. Amikor az `fdisk` alkalmazást futtatom, a változtatások nem lesznek tartósak, még a számítógép újraindítása után sem. A SCSI illesztőkártya érzékeli a merevlemezt, és az alacsony szintű formázás és ellenőrzés hiba nélkül fut le. Mégis, amikor az `fdisk` programot használom, a lemezrész létrejön ugyan, de a változtatás nem történik meg ténylegesen.
Eskinder Mesfin, mesfin@attbi.com

Úgy tűnik, hogy a változásokat nem írod ki a lemezfelosztási táblázatba. Az `fdisk` addig nem ír a lemezre, amíg nem utasítod rá. Mielőtt a `q` paranccsal kilépnél, a változtatások kiírásához add ki a `w` parancsot.
Christopher Wingert, cwingert@qualcomm.com

Az SSH nem enged be

Telepítettem egy Red Hat 7.3-as és egy Red Hat 8.0-s rendszert. Amikor SSH-t szeretnék használni, hibaüzenetet kapok: *Connection Refused*. Mindkét gépen be tudok lépni SSH-val saját magára, de a másik gépről ez nem megy. Leállítottam az IP Tables szolgáltatást, és ellenőriztem, hogy a */etc/xinetd.d/telnet* állományban a `disable=no` sor jelen van-e. Kiadtam a `netstat -t | grep telnet` és a `netstat -t | grep ssh` parancsokat, amik arról tájékoztattak, hogy mindkét szolgáltatás fut.
Robert Haack, haack@nclack.k12.or.us

Ellenőrizd, hogy tudod-e pingelni az egyik gépet a másikról. Ha igen, akkor az `nmap` programhoz hasonló eszközzel ellenőrizheted, hogy a megfelelő kapuk nyitva vannak-e a másik gépről nézve.
Chad Robinson, crobinson@rfgonline.com

Nézd meg a */var/log/messages* vagy a */var/log/auth.log* naplóállományt. Ezekből ki kell derülnie, hogy miért dobja el az SSH a kapcsolatot. A számítógéped valószínűleg a fordított DNS-lekérdezést próbálja végrehajtani az IP-címedhez, és ez nem megy. Ezt a hibát úgy lehet kiküszöbölni, hogy a */etc/hosts* állományt feltöltöd a számítógépeid IP-címeivel és névadataival.
Marc Merlin, marc_bts@google.com

Ellenőrizd, hogy a */etc/hosts.allow* állományban szerepel-e az `sshd: ALL` (vagy a távoli gép IP-címe) sor, mert valószínűleg ezért nem tudsz belépni a másik gépről.
Mario Bittencourt, mneto@argo.com.br

Az SSH mindent tud, amit a Telnet, sőt még többet is. Titkosítást is használ. Emiatt javaslom, hogy az elavult Telnet szolgáltatást kapcsold ki. Különösen napjainkban, amikor már mindenhol vezeték nélküli hálózatok vannak, nem kockáztathatod, hogy felfedd a jelszavadat a hálózaton. Jól tetted, hogy a `netstat` programot használtad a figyelő SSH-démon ellenőrzésére, bár a `-a` kapcsolót még hozzá kell tenned. Add ki ezt a parancsot:

```
$ netstat -at | grep ssh
```

Keresd meg az ehhez hasonló sorokat:

```
tcp   0 0 *:ssh   *:*    LISTEN
```

Ebből látszik, hogy az `sshd` várja a bejövő kapcsolatokat.
Don Marti, dmarti@ssc.com

Ha az SSH-démon nem fut, indítsd el az `sshd start` szolgáltatással, vagy állítsd be, hogy a rendszerindítás után önműködően elinduljon:

```
chkconfig --level 2345 sshd on
```

Felipe E. Barousse Boué, fbarousse@piensa.com

Linux Journal 2003, 107–109. szám

Új termékek

PRISMIQ MediaPlayer

A PRISMIQ MediaPlayer egy tévé tetejére helyezhető doboz, ami képes a filmeket és zenéket a tévén keresztül a számítógépről lejátszani, valamint széles sávon csatlakozik az internetre is. A PRISMIQ készülékben 32 bites NEC uPD61130 MIPS processzor, beépített MPEG-dekódoló, 16 MB Flash ROM és 64 MB SDRAM található. A 10/100-as ethernethez való RJ45-os csatlakozó és egy cardbus/PCMCIA-foglalat a vezeték nélküli kártyához gondoskodik a hálózati kapcsolatról. A készülék az MPEG-1 és MPEG-2 videoformátumokat, az MP3 hangformátumot, valamint a JPEG, GIF és PNG képformátumot támogatja. A kimeneti csatlakozók: S-video, kompozit video, S/PDIF, RCA-audio (jobb és bal oldal sztereó). A MediaPlayer a Linux 2.4-en alapul és webböngésző programot is mellékelnek hozzá. A vezeték nélküli billentyűzet választható.
Adatok: PRISMIQ, 2121 South El Camino Real, 10th Floor, San Mateo, California 94403,
e-mail: sales@prizmiq.com,
➲ http://www.prismiq.com

SnapGear PCI630

A SnapGear PCI630 egy VPN-tűzfalat megvalósító PCI-kártya, ami minden VPN-nel kapcsolatos terhet levesz a gazdaszámítógép válláról, továbbá távolról beállítható, nagy biztonságot tesz lehetővé és egyszerűen telepíthető. A PCI630 egy hálózati kártya helyét elfoglalva nyújtja a biztonságos elérést és kapcsolattartást több VPN-csatornán. Kiszolgálókba vagy munkaállomásokba egyaránt beszerelhető, 10/100-as ethernetkapcsolattal, 4 MB flashmemóriával és 16 MB RAM-mal rendelkezik. Támogatja a legfeljebb 2048 bites RSA kulcsok alapján történő hitelesítést, az X.509 tanúsítványokat DER és PEM formátumban, valamint több alhálózatot; mindezt egy harmadik fél által gyártott ügyfélprogram szükségessége vagy a felhasználók száma szerinti korlátozás nélkül.
Adatok: SnapGear, Inc., 7984 South Welby Park Drive #101,
West Jordan, Utah 84088,
e-mail: contact@snapgear.com,
➲ http://www.snapgear.com

AMD Opteron processzorok

Az AMD bejelentette, hogy nyolcadik nemzedékbeli vállalati osztályú processzormagja, az Opteron 64 bites x86 megoldást fog alkalmazni. Az Opteront olyan sok fizikai és virtuális memóriát igénylő alkalmazások futtatására tervezték, mint például a nagy teljesítményű kiszolgálók, adatbázis-kezelő rendszerek és a CAD-eszközök. A meglévő 32 bites és az új 64 bites kódokat is nagy teljesítménnyel hajtja végre. Az Opteronba épített HyperTransport, azaz nagy sebességű közvetlen kapcsolat az integrált áramkörök között csökkenti a B/K műveletek miatti fellépő szűk keresztmetszetet, növeli a sávszélességet és csökkenti a késleltetést. A beépített memóriavezérlő a memória elérésének szűk keresztmetszetét csökkenti.
Adatok: AMD, PO Box 3453, Sunnyvale, California 94088,
➲ http://www.amd.com

Red Hat Enterprise Linux ES és WS

A Red Hat két új vállalati környezetbe szánt termékkel jelentkezett, amelyek csereszabatosak a Red Hat Enterprise Linux AS – régebbi nevén a Red Hat Advanced Server – operációs rendszerrel. A Red Hat Enterprise Linux ES a belépő szinttől a vállalati osztály szintjéig jelent megoldást a hálózati, fájlkiszolgáló, nyomtatási, levelezési, web és egyéni vagy „dobozos" kereskedelmi alkalmazások futtatására. Ezt az operációs rendszert kisebb rendszerekre tervezték, amelyekben legfeljebb két processzor és 4 GB memória van. Kétféle változatban kapható: Basic és Standard. A Red Hat Enterprise Linux WS egy olyan mérnöki munkaállomás, amelyet kiszolgálóalapú, illetve programfejlesztői környezetbe terveztek független programgyártók ügyfélalkalmazásai számára. A WS is Standard és Basic változatban kapható, és legfeljebb kétprocesszoros munkaállomásokat támogat.
Adatok: Red Hat Software, 2600 Meridian Parkway, Durham, North Carolina 27713,
➲ http://www.redhat.com

Visual SlickEdit 8.0

A SlickEdit, Inc. megjelentette a Visual SlickEdit programozóknak szánt szövegszerkesztő 8.0-s változatát. A Visual SlickEdit 8.0 többféle kódszerkesztő eszközt tartalmaz, amelyek több programnyelvet és karakterkódolást támogatnak.
A 8.0-s változat újdonsága a könyvtáralapú projektek kezelése, az osztott címkék önműködő frissítése, biztonságos FTP (SFTP), valamint kisegítő lehetőségek vak és csökkent látású fejlesztők számára. Új háromirányú összehasonlító felülettel bővült a DIFFzilla fájl- és könyvtárösszehasonlító motorja.
A 8.0-s változatban tovább nőtt a Jbuilder, Java, GNU C/C++ és a CVS támogatása. A termék honlapjáról letölthető az egy hónapig használható ingyenes próbaváltozat.
Adatok: SlickEdit, Inc., 3000 Aerial Center Parkway, Suite 120, Morrisville, North Carolina 27560,
➲ http://www.slickedit.com

Eventide VR778

Az Eventide VR778 egy olyan digitális hangfelvevő és hangarchiváló rendszer, amit például segélyhívó rendszerekben (mentők, rendőrség stb.) lehet használni. Az eszköz az előlapján elhelyezkedő grafikus képernyő segítségével vezérelve önállóan is használható, de hálózaton keresztül számítógépről is elérhető.
A VR778 hibatűrő készülék: két menet közben cserélhető tápegység, hűtőrendszer és két merevlemez van beépítve. A VR778 egyszerre 8–160 analóg és 16–96 digitális csatorna jelét képes felvenni és keverni. A felvétel többféle mértékben tömöríthető: 13,3, 16, 32 és 64 Kb/s közül lehet választani. A két 120 GB-os merevlemez RAID-1 módszerrel van tükrözve (választható a 380 GB-os RAID-5 elrendezés is), és legfeljebb 19 800 óra hangot tud felvenni a 13,3 Kb/s tömörítés mellett. Az eszközbe DVD-meghajtókat meghajtókat is építettek.
Adatok: Eventide, Inc., 1 Alsan Way, Little Ferry, New Jersey 07643,
➲ http://www.eventide.com

Linux Journal, 2003, 110. szám

Az elsorvadt PC-forradalom új hajtóereje

Kemény küzdelembe kerül, hogy a Linux eljusson az üzletek polcaira.

A személyi számítógépek elsorvadt forradalmának idejét éljük. Biztos vagyok benne, hogy sokan vonnák kétségbe ezt az állítást, tekintetbe véve azt az óriási hatást, amit a számítógép a kapcsolattartás, az üzletkötés és a szórakozás módjaira gyakorol. De amikor körülnézek, azt kell kérdeznem magamtól: miért nem látok még több személyi számítógépet még több helyen? Miért nincs számítógépe minden diáknak? Miért nem kapnak az ötévesek saját számítógépet ugyanúgy, ahogyan biciklit kapnak? Miért nincs számítógép minden szállodai szobában, konferenciahelyiségben, tanteremben és a lakások minden szobájában? Miért rendelkezik az amerikaiak csupán hetven százaléka PC-vel?
A legfőbb válasz természetesen az, hogy a számítógépek túlságosan drágák. Számos fogyasztó számára a személyi számítógépek legdrágább összetevője nem maga a gép, hanem a programok ára. Az átlagos PC-k 700 dolláros áron kelnek el, és ez évente több mint száz dollárral csökken – ez az átlag. Sokan lényegesen kevesebbet fizetnek. De a programok ára a nullához közelítő sokszorosítási költségek ellenére sem mutat olyan látványos csökkenést, mint az alkatrészáraké. A valódi verseny hiánya lehetővé tette, hogy egy cég 85%-os árréssel gazdálkodjon, ami nagyszerű a részvényeseknek, de dollármilliárdokkal emeli meg az informatikai költségeket, és megakadályozza, hogy a számítógépek előnyeit társadalmunk minden szegletében kihasználhassák. A monopóliumellenes perek és a kormányprogramok kevés sikerrel próbálták leszorítani a költségeket és csökkenteni az informatikai szakadékot.
A személyi számítógépeken használt Linux lesz az a hajtóerő, ami a PC üzletág következő fellendüléséhez energiát szolgáltat, és ehhez képest az előző fellendülés, a grafikus felület elterjedése szerénynek fog tűnni. A programok ára meredeken esni fog. Ahelyett, hogy az informatikai költségek nagyobbik felét felemésztené, a programok ára sokkal kisebb részt tesz majd ki, körülbelül annyit, amennyit a merevlemezért és a memóriáért fizetünk.

Jó néhány akadályt kell még leküzdeni, mielőtt a felhasználói oldalon a Linux lényeges hatással lehet a programárakra, és az akadályok legtöbbje nem műszaki jellegű. Amikor alkalmam nyílik rá, hogy felhasználókat ültessek le egy jól beállított, Netscape-pel, StarOffice-szal és KDE-vel ellátott Lindowst futtató gép elé, általános reakciójuk a megdöbbenés. Megdöbbenti őket, hogy a Linux milyen jól használható a személyi számítógépeken. Van még mit tenni egy könnyen használható rendszer és a sokszínű programválaszték kialakítása érdekében, amik együttesen biztosítják a megfelelő felhasználói élményt. A Lindows OS Click-N-Run rendszer, úgy érzem, kezdi betölteni ezt az űrt.
A Linuxnak a műszaki területen kívül kell felnőnie, mielőtt széles körben elfogadják a felhasználók. Itt pedig a fő akadály az, hogy a bebetonozott monopóliumhelyzetben lévő cég és annak pénzeszsákja miatt a terjesztői hálózat minden lényeges szintje úgy épül fel, hogy széllel szemben kell hadakoznunk. Nagy kihívás rávenni az összeszerelő cégeket, hogy linuxos gépeket építsenek, amikor nyereségességükre a már említett monopóliumhoz fűződő gazdasági kapcsolatuk van a legnagyobb hatással. A tíz legnagyobb gyártó közül nem egy jelezte, hogy nem telepíthet addig Linuxot a személyi számítógépeire, amíg nem „tisztázzák az ügyet” – és ezt nem úgy kell érteni, hogy a belső cégvezetéssel tisztázzák. Ha a gyártókat megnyertük, a következő kényes lépés értékesítőket találni a termék számára. Sokan tesznek föl kérdéseket a kereslettel, az eladók képzésével és a felhasználók támogatásával kapcsolatban, de ezekre a kérdésekre – amint a pénztárgép csilingelni kezd – kedvező válaszokat találnak.
Ha a linuxos gépeket értékesítő internetes áruházak, mint a Walmart.com vagy a Tigerdirect eladásai alkalmasak valamiféle előrejelzésre, akkor ezek szerint erős kielégítetlen kereslet létezik, ami azt jelenti, hogy különösen a korán ébredő kereskedők jelentős forgalomra számíthatnak. A Brick, Kanada legnagyobb elektronikai áruházlánca több mint ötven üzletében kezdett gépeket árulni előre telepített Linux-rendszerrel. Ez óriási előrelépés, hiszen 15 éve először van választási lehetőségünk egy számítástechnikai üzletben.
Miután a Linux széles körű elterjesztéséhez szükséges értékesítési rendszer ígéretesen fejlődésnek indult, az utolsó akadály a képzés. A legtöbb felhasználó csak a Microsoftot ismeri, csak ennek a számítástechnikai nyelvét beszéli.
A Microsoft még a történelem újraírásával is megpróbálkozik, amikor saját találmányának tekint még egy olyan alapvető kifejezést is, mint a „windows” (ablakok) – és a saját tulajdonaként kezeli. Az azonban nem tagadható, hogy a Microsoft fájlformátumai és protokolljai de facto szabvánnyá váltak, és ezekkel minden felhasználói programnak zökkenőmentesen együtt kell működnie. A mai linuxos programok meglepően nagyszerű teljesítményt nyújtanak ezen a téren, de a fogyasztók nem tudnak erről. Ez az a terület, ahol a munka neheze még hátravan. Konferenciák, igehirdetők, képzések, Linux-sor az üzletekben, „Lin” címke a „Mac” és a „Win” mellett az alkatrészeken és még sok minden más szükséges a tanulási folyamat felgyorsításához. Végül a linuxos gépek előnyös áruk hatására elterjednek az üzleti életben, a háztartásokban és az iskolákban, de a növekedés ütemét a tanulási folyamat sebessége diktálja. Alig várom azt a napot, amikor a fogyasztók a kedvező árú Linux-programok előnyeit élvezhetik, mivel a mai állapotnál sokkal nagyobb lesz az a kedvező hatás, amit a számítógép az életünkre gyakorol. Nemsokára eljön ez az idő, de ha ráveszed egy barátodat egy linuxos gép kipróbálására, még hamarabb beköszönt.

Linux Journal 2003. június, 110. szám

Michael L. Robertson
A Lindows.com alapítója és ügyvezetője.

Click-N-Run: egyszerűbb jövő a vásárlóknak?

A számítógépek egyszerű kezelhetőségének fennmaradt akadályai nem technikai jellegűek, hanem a kereskedők programterjesztési módszereiben keresendők. A Click-N-Run szolgáltatás megkönnyítheti mindezek leküzdését.

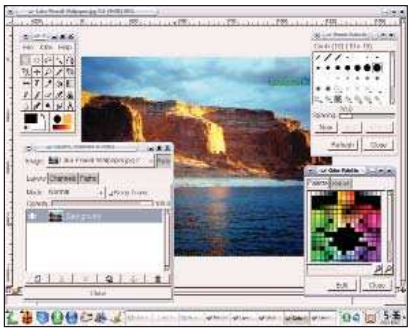
Idén februárban vonatra szálltam, hogy otthonomból, a kaliforniai Santa Barbarából eljussak a szintén kaliforniai San Diegóban rendezett Desktop Linux Summit elnevezésű tanácskozásra. Megközelítőleg 50 dollárt fizettem egy első osztályú oda-vissza útért, ami kevesebb volt, mint a kocsimba a benzin, kényelmesebb, mint egy első osztályú repülő; és az ülésem még váltakozó árammal is el volt látva hordozható számítógépem számára. Az út legnagyobb része paradicsomi tájon keresztül vezetett: smaragdzöld szántók, vörös kanyonok, elővárosok, amiket hatalmas hófödte hegyek ölelnek körül.

Gondolatok a vonaton

Az utazás legizgalmasabb része azonban – legalábbis számomra – az az óriási ipari övezet volt, ami a San Fernando-völgyben kezdődik, és a Los Angeles folyó betoncsatornáján keresztül egészen Los Angelesig és az Orange megyei síkságokig terjed több száz négyzetmérföldön keresztül. Az ablak előtt végtelennek tűnő rönkfák sora, mindenféle méretű csövek és fémáruk sokasága suhant el. Számomra mindez élő példája a számítógépes iparág jövőjének, amiben ezek az árucikkek jó és nélkülözhetetlen dolognak számítanak.
A számítástechnikai ipar rövid története során az ipar nemkívánatos dologként tartotta számon a programot, ami nagyon alacsony vagy nulla haszonkulccsal bír. Az érett iparágak azonban mégis szép hasznot húznak az alapanyagokból. A fakitermelő és bányavállalatok, a fémfeldolgozó, műanyag fröccsöntő cégek mind a nyersanyagüzletben érdekeltek. Amikor a programipar felnő, akkor fogja csak megérteni a nyersanyagok értékét, elsősorban annak a nyersanyagnak az értékét, amit mi szabad programnak hívunk. Szabad nem abban az értelemben, mint az ingyen sör vagy a szólásszabadság, hanem ingyenes, mint a mészkő, a fa vagy a szilícium. Ezeket az alapvető elemeket a természet szabadon hozza létre. Hasonló módon, mint ahogy az emberi elme előállítja a szabad programot.
Nagyon nehéz meglátni a szabad programok gazdasági hasznát egy olyan iparágban, amit még mindig egy óriási mutánsvállalat ural, ami monopolhelyzetét arra használja, hogy 85%-os árrést csikarjon ki a vevőktől, akiknek vajmi kevés választásuk van az ügyben.
Hiszem, hogy a diegói Desktop Linux Summit tanácskozáson egy érett programipar hajnalának voltam a tanúja. Mielőtt odaértem volna, nem voltam egészen biztos benne, hogy mit is várok a konferenciától. A kiállítás megrendezése nem volt felhőtlen. Bizonyos kapcsolattartási zavarok léptek fel a konferencia szervezői csoportjai között. Néhány előadó és cég visszalépett. Az egyetlen, amivel mindenki egyetértett, hogy a kiállítás majdnem annyira a Lindowsról szólt, mint a Linuxról. Miután lehetőségem nyílt arra, hogy lássam, mivel rukkol elő a Lindows, úgy gondolom, ez így volt helyes.

A Lindows a jövő?

Ha a Lindows sikeresen teljesíti küldetését, amire úgy gondolom, nagyon jó eséllyel pályázik, a munkaasztali Linux végre valósággá válik, s nem változik pusztán egy csaknem üres gödörré a kiszolgálók és a beágyazott eszközök mellett, ahol is a Linux a legjobb úton halad a világuralom elérésében.
Miközben a Dell, a Gateway, a HP és az IBM mind a fenekükön ülnek és a piac segélykiáltását várják, hogy vegyék már komolyan a munkaasztali Linuxot, addig a Lindows.com keményen dolgozik a piac kialakításán. Ez nem egyszerűen csak egy kellemes, új terjesztést jelent, hanem új üzleti modellt is, aminek révén mind a szabad programok, mind a kereskedelmi programok győztesek lesznek.
A Lindows.com ***Michael Robertson*** teremtménye. Ő az, aki megalapította, majd később elnöke és ügyvezetője lett az MP3.com-nak. Robertson egyszerűen visszavonulhatott és gondtalan életet élhetett volna, kezelhette volna a portfólióját, miután a Vivendinek eladta az MP3.com-ot. Ehelyett elhatározta, hogy olyasmit talál ki, amit egy kockázatitőke-befektető sohasem tett volna meg: egyenesen az operációs rendszerek piacán fog versenyre kelni a Microsofttal.



Miközben mások a Microsoft sikerét kihívásnak látják, Robertson puszta üzleti lehetőséget lát benne, s ez bizonyos értelemben lázadó jellemre vall. A következőket nyilatkozta a konferencián, amikor interjút kértem tőle: „Jelenleg azt látjuk, hogy a gépkereskedők hét százalékos bruttó nyereségért ölik egymást. Szörnyű üzlet. Mindeközben a Microsoft fölözi le saját magának a nagy árréssel realizálható programok hasznát. Itt állunk ebben az öldöklő üzletben, gyártjuk a borotvanyeleket, miközben a Microsoft a borotvapenge előállításával az összes profitot elviszi.

Mire való a marketing?

A jövőben mindez meg fog változni. A gépgyártó cégek partnerséget fognak kötni az olyan vállalatokkal, mint amilyen a Lindows.com, ami mindkét fél kapcsolatának fenntartásában érdekelt. Azt fogják mondani: „Jó napot, hajlandó vagyok befekteti a PC-ket gyártó cégekbe, ezeket fogom a piacra vinni, árulni és támogatni. De én olyan partnereket szeretnék, akik a puszta gépért árrésnél többet tudnak nekem adni. Bármilyen borotvapengét árulsz is a vevőknek – vírusvédelmet, webszűrést, levelezőszolgáltatást, bármit –, én ennek szeretném egy részét." És mi készen állunk, hogy ezt nyújtsuk nekik.

Nem egyszerűen arról beszélek, hogy „adj hozzá egy leltározási egységet az online webáruházadhoz". A Walmart.com-mal ezt mi már megvalósítottuk. Én a valódi marketingről

beszélek, amit nagyban lehet csinálni, és ami majd kihúzza ezeket a nagy gépgyártó vállalatokat ebből a siralmas helyzetből. Mindannyian nyersanyagot árulnak, ami azt jelenti, hogy a legszikárabb, legegyszerűbb, legalacsonyabb költséggel dolgozó szállító lesz a győztes. Ez pedig a Dell. A másik oldal – a mai vesztesek – pedig azt fogják mondani: „Változatnunk kell egy picit a modellen, mégpedig úgy, hogy jobb megállapodásokat kötünk a partnereinkkel, akiknek jobb értékesítési modelljük van, és a saját operációs rendszereiket használják."

Ezt a jobb modellt Click-N-Runnak hívják. Ezt építették bele a LindowsOS-be, ami Debian GNU/Linux-, valamint KDE-alapokon nyugszik.

Tekintsd a Lindows OS-t egy olyan rendszernek, amit saját magad alakíthatsz ki a maximális felhasználói kényelmet figyelembe véve, és nincsenek hagyományos programfelhasználási-szerződési és jogdíjgondjaid sem, amikkel törődnöd kellene. Robertson célja az egyszerű kezelhetőség, ami nyilvánvalóan felette áll a Windowsnak, különösen akkor, ha az olyan nehezebb részekhez érkezünk, mint például az új programok telepítése. Ez az amiről végeredményben a Click-N-Run szól. Az új HP nyomtatód számára szeretnél új meghajtót? Kattints a letöltés hivatkozásra, a LindowsOS pedig elindítja a telepítőt, ami felhozza a Konqueror böngészőt. Ezek után már csak egy-két utasítást kell követni, és nemsokára nyomtathatsz is LaserJet nyomtatódon. Hála a Debian függőségi modelljének, a nyomtató hozzáadása nem tesz tönkre mást. Egy asztali planetáriumot, a KStarsot szeretnéd használni? Semmi izzasztó gyakorlat: kattints rá, és a tiéd lehet. Gimp? Természetesen. Kattints rá, és futtasd!

Ezek azonban mind szabad programok. A Click-N-Run ugyanezek között a keretek között vegyíti a kereskedelmi programot is. Ha a Marble Blastot szeretnéd használni, akkor ez 9,95 dollárba fog kerülni. Létezik egyéves előfizetési díj is; ezért az összegért azt kapod, amit a Microsoft vagy az Apple sohasem adott meg: igazi kapcsolatot az operációs rendszer szállítójával.

A régi programipar csak a termelésről szólt. Előállítasz egy terméket, az ellátási láncon keresztül eljuttatod a felhasználóhoz, és a sikert negyedéves eladási eredményeken méred le. Bármilyen kapcsolatod is legyen a végfelhasználóval, az távoli és közvetett. Ha valódi kapcsolatot keresel, az nem mindig kölcsönös és bizalmi. A legtöbb esetben kizárólag arról akarsz meggyőződni, hogy a vásárló a termékednek nem egy kalózmásolatát használja-e.

A Click-N-Runnal a kapcsolat kölcsönös. Ha ez egy bizalmi kapcsolat, akkor a Lindows.com közvetítőként léphet fel az általa szállított szabad és kereskedelmi programok gyártóinál. Ahelyett, hogy azt kérdeznék, „hogy szerepel az 1.04-es változatunk a magas jövedelemmel rendelkező, keleti parton élő vásárlóink körében?", a következőket kérdezik: „hányan töltöttek le programokat a játék kategóriában?, vagy „mennyien töltötték le a Gimp bővítményt?". Ha tízezer ember töltötte le az AbiWordöt, és tisztán látszik, hogy jelentős részük új szolgáltatást szeretne az AbiWordbe, a Lindows.com vagy az egyik partnere talán anyagilag is támogatni fogja ennek a szolgáltatásnak a fejlesztését. Mindezt annak ellenére tenné, hogy ők magán a programon nem fognak pénzt keresni.

Véleményem szerint a Click-N-Runnal a Lindows.com-nak megvannak az eszközei arra, hogy a világ legvevőérzékenyebb programgyártó vállalatává váljon, s mindezt olyan módon tegye, hogy mindenkit hozzásegít a győzelemhez.

S hogy milyen közel jutottak ehhez az eszményképhez? A Lindows.com egy 799 dolláros hordozható számítógépet is piacra dobott a kiállítás alkalmával, ami csak 1,3 kg-ot nyom. Az egyik résztvevő a „rendszergazda PDA-jának" nevezte el, ami igaz is. Hatéves fiam szintén beleszeretett. Már olvas, és körülbelül 10 másodperc alatt megtanulta kezelni a Click-N-Runt. Ezután letöltött egy csomó játékot, és ezekkel játszott a vonaton egészen hazáig. Otthonunkban háromféle felületen futnak a gépek. Találjátok ki, melyiket szereti a legjobban a gyerek!

Linux Journal 2003. június, 110. szám

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

Megvilágításmodellezés a Radiance-szal

Vázlattól a leképezett képig – néhány óra alatt.

Amikor egy rönkházat szerettem volna megtervezni a számítógépemen, hogy lássam, milyen látványt nyújt majd bizonyos fényviszonyok között, először az AutoCAD-del, a 3D Studio Maxszal és számos egyéb otthontervező kereskedelmi termékkel próbálkoztam. Egyik sem nyújtott valóságosnak ható kimeneti képet, vagy az általam elképzelt rönkfalak kezeléséhez valamilyen könnyen használható módszert. Ezután eljátszottam a Lawrence Berkeley National Laboratory (LBL) Radiance nevű világításszimulációs programjával, és arra a megállapításra jutottam, hogy ha segédprogramokkal egészítem ki, ezzel sokkal gyorsabban érek célba.

De mi is az a Radiance?

A Radiance egy eredetileg *Greg Ward Larson* által írt megvilágításszimulációs program. Már a kilencvenes évek elejétől hozzáférhető, és a nem üzleti felhasználók számára adott ingyenességet mostanában váltotta fel a nyílt forrású felhasználási szerződés. A programcsomag az előállított igen látványos képeket egy különleges formátumban tárolja, ami egyaránt rögzíti az elrendezés felületi mintáit és a megvilágítás körülményeit, az Autodesk LightScape és VIZ 4 profi felhasználásra készült programjaihoz hasonlóan. A film- és játékkészítésben használt programcsomagok a gyorséttermekben kapható ételekhez hasonlíthatók: a végeredmény lehet vonzó és népszerű, de a valósághoz nem sok köze van. A film- és játékkészítők számára a világítás fizikai részletei egyszerűen nem olyan fontosak, mint a sebesség, hiszen nagyon sok képpont gyors kezelését kell megvalósítani: egy kétórás film 172 800 képkockából áll, a játékoknak pedig valós időben kell futniuk. Az eredmény az, hogy a grafikus rendszerek által előállított kép inkább egy művészi algoritmus eredménye, ami kevéssé támaszkodik valóságos alapokra.
A Radiance kimenete a fizikai fény olyan laboratóriumi hűségű utánzása (már amennyire a bemenet valóságos), ami a szakmai közönség szigorú próbáin megállta a helyét.

A Radiance telepítése

A Radiance forráskódja a ➲ http://radsite.lbl.gov címről szerezhető be. A forráskód letöltését javaslom az előrefordított RPM-csomagok helyett, ugyanis az RPM-ek a bővítményeket magukban foglaló állományok egyikét sem tartalmazzák. Ha sikerült letöltenünk a tarcsomagot, teendőnk a következő:

```
$ tar xzf rad3R4.tar.gz
$ cd ray
$ ./makeall install
```

Ezután válaszoljunk arra a kérdésre, hogy hova szeretnénk elhelyezni a programot. Én a ***$HOME/radiance/bin*** könyvtárat használtam a bináris állományok, és a ***$HOME/radiance/lib*** könyvtárat a bővítmények tárolására.
A `makeall` parancsfájl nem telepíti a leírást és a példaképeket, ezek számára is nekünk kell valami alkalmas helyet találnunk, például:

```
$ mv doc/man $HOME/radiance
$ mv obj $HOME/radiance
```

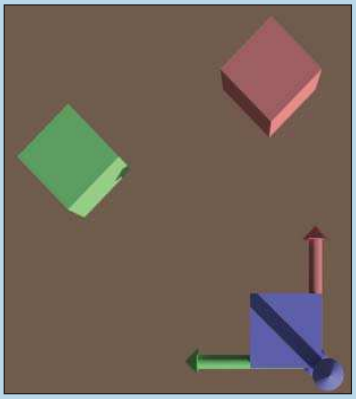
Gondoskodjunk róla, hogy ezek az elérési útvonalak szerepeljenek a saját profilunk `MANPATH` és `PATH` változóiban. Egy figyelmeztetés: létezik a csomagban egy fontos segédprogram, amit ***rview***-nak hívnak. Sajnos a Vimben szintén van egy ugyanilyen nevű segédprogram, úgyhogy vagy meg kell változtatnunk a `PATH` változót, vagy a Vim `rview` programját kell átneveznünk. A Radiance segédprogramjának ne adjunk új nevet, mert a programcsomag más segédprogramjai közvetve hivatkoznak rá.

A Radiance bemenete

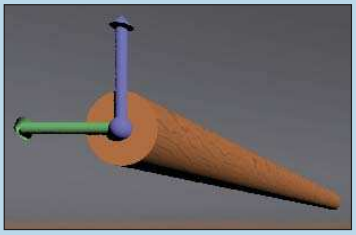
A Radiance új felhasználóinak először a jelenetek megalkotására szolgáló belső CAD-rendszer hiánya szúr szemet. A programot a 90-es évek elején kutatási céllal írták Unix alá, és ha a fájlformátumot megvizsgáljuk, az is azonnal nyilvánvalóvá válik, hogy a rendszer a hozzám hasonló, a csővezetékek és a szöveges folyamatok hatékonyságától elragadtatott parancssormegszállottaknak készült (elvégre a monogramom is AWK). Mindamellett rendelkezésre állnak azok a segédprogramok, amelyekkel a DXF, Wavefront, MGF vagy az ezekhez hasonló formátumú geometriai leírásokat lefordíthatjuk, így bármilyen programot használhatunk, ami képes az ilyen formátumba történő mentésre. A ➲ http://linux.org címen felsorolt modellezőprogramok jelentős része képes ilyen kimenet előállítására. A Desktop Radiance nevű Windows alapú AutoCAD/Radiance modul a Radiance honlapjáról is letölthető, ha éppen az AutoCAD megfelelő változatával rendelkezünk.
A Radiance bemeneti fájljai olvasható szövegek, amelyek remek lehetőséget nyújtanak parancsfájlok létrehozására. De legyünk óvatosak: a leírás helyenként olyan kifejezéseket tartalmaz, amiktől a fizikusok vérnyomása a magasba szökken, mint például a „watt per négyzetméter per térszög”. Nem árt a honlapon lévő összes leírást áttanulmányozni. Ha nem csak játszadozni szeretnénk a programmal, talán érdemes beszerezni Greg Ward Larson „Rendering With Radiance” (Fényleképezés a Radiance használatával) című könyvét, vagy valami hasonlót. A könyv pillanatnyilag nem kapható, az antikváriumokban érdemes utánanézni (amerikai adat – a szerk.).

Saját próbálkozások: egy mintajelenet megalkotása

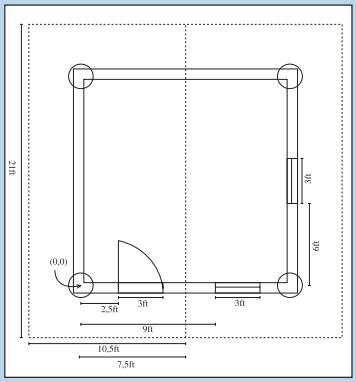
Az 1. lista egy eget és földet ábrázoló kép, amin az anyag beállítása réz, és ebből a rézből egy gömb helyezkedik el a képen. Az ég és a föld nincs külön beállítva. Az egyetlen, amit a saját képhez meg kell változtatnunk, azok a `gensky` parancs beállításai. A listában lévő értékek november 25 déli időpontjának felelnek meg az északi szélesség 25. és nyugati hosszúság 80 fokán. A déli és keleti elhelyezkedést negatív értékekkel adhatjuk meg. Az első sor a tételre vonatkozó, már létező anyagot határozza meg (illetve üres részt, ha nincs megadva), az anyag vagy geometriai alakzat fajtájának a nevét (sphere – gömb, polygon – sokszög, plastic – műanyag, metal – fém),



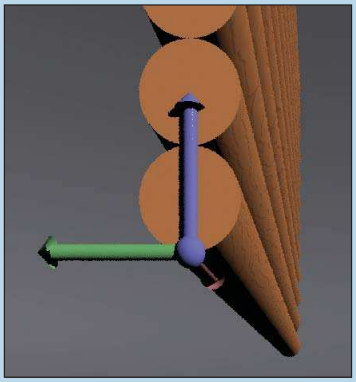
1. kép



2. kép

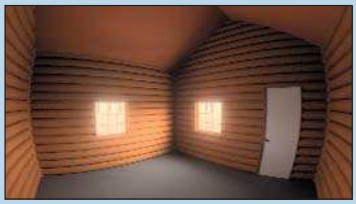


3. kép



4. kép

5. kép



6. kép

1. kép
Az xform viselkedése. A koordinátatengelyek színkóddal rendelkeznek: +x = vörös, +y = zöld, +z = kék. A kék kockát változatlanul hagytam. A piros kockát elforgattam, majd eltoltam. A zöld kockát először eltoltam, ezután forgattam

2. kép
A genlog módosítatlan kimenete. A tengelyek VZK/xyz sorrendűek (V = +x, Z = +y, K = +z), az x tengely a rönk belsejében van

3. kép
Egy házikó alaprajza. A nyílások méreteit annak a falnak a kezdőpontjaitól kell megadni, amelyen elhelyezkednek

4. kép
A genlogwall kimeneteként született, nem átalakított fal. A tengelyek színsorrendje: VZK/xyz

5. kép
A házikó nappali perspektivikus képe, néhány fával az érdekesség kedvéért. A kiadáshoz szükséges 3300×2200 minőségű leképezés egy 1,7 GHz-es noteszgépen körülbelül öt órát vett igénybe

6. kép
A házikó belsejének nagy látószögű képe külső napfény-megvilágítással. A kiadáshoz szükséges 3300×2200 minőségű leképezés egy 1,7 GHz-es noteszgépen körülbelül öt órát vett igénybe

és egy, a felhasználó által megadott nevet. A következő három csoport a tétel karakterlánc egész és valós (lebegőpontos) típusú értékei. Minden sor az értékek számával kezdődik, ezt követik maguk az értékek, amelyek annyi sort foglalhatnak el, amennyi szükséges.
A legtöbb bejegyzés csak valós értékkel bír. Ez a magyarázata a bejegyzések közepén gyakran előforduló két nullának: nincs karakterlánc vagy egész típusú érték. A réz (brass) utolsó sorában lévő 5-ös szám azt jelzi, hogy öt valós érték következik, a gömb (sphere) utolsó sorában lévő 4-es pedig négy valós értéket jelez. Az értékek meghatározott sorrendben követik egymást, például a gömb megadásához egy középpontra (x; y; z) és a sugárra van szükség.
Gyakran az anyagok jelentik az elrendezés legnehezebb feladatát. A legegyszerűbb, ha már létező anyagokat másolunk, és ezeket módosítjuk az igényeinknek megfelelően. Erről további részleteket a honlapon elérhető ***refman.pdf*** fájlban találhatunk.
Az 1. lista (49. CD Magazin/Radiance könyvtár) tetején lévő `gensky` egy beágyazott parancssoros segédprogram. A sor elején lévő felkiáltójel arra utasítja a rendszert, hogy a sort egy olyan héjparancsnak tekintse, aminek a kimenete a kép részét alkotja. A Radiance számos ilyen segédprogramot tartalmaz, és tapasztalatom szerint a sajátjaink megírásával jóval egyszerűbbé és gyorsabbá válik a jelenetek megalkotása.

Az elemek mozgatása

A legtöbb jelenetszerkesztő program az elemeket a kezdőpontban helyezi el, ami valószínűleg nem egyezik meg a kívánt helyzettel. Ezen a gondon az `xform` segédprogrammal tudunk segíteni, aminek a parancsformátuma a következő:

```
xform -t transx transy transz
      -rx angle
      -ry angle -rz angle
      -s scalefactor tetszőleges_jelenetfájl
```

Az `xform` bármit fájllá tud alakítani, és lehetőségünk nyílik arra is, hogy ráirányítsuk egy jelenetszerkesztő program kimenetét. A program alkalmas az objektumok átméretezésére (`-s` szorzótényező), tengely körüli forgatásra (`-ry` forgatási_szög az y

2. lista A nyílásleíró fájl (holes/southwall.holes) a déli fal számára

```
# Bejárati ajtó (padlótól 7 láb, kezdet:
# 2,5 láb a fal kezdőpontjától,
# 3 láb szélesség)
0 7 2.5 3:d
# ablak (3 lábtól 6 lábig a padlótól,
# kezdőpont: 9 láb a faltól, 3 láb széles)
3 6 9 3:w
```

3. lista
A nyílásleíró fájl (holes/eastwall.holes) a keleti fal számára

```
# keleti fali ablak (3-6 láb a padlótól,
# kezdőpont: 6 láb a fal kezdőpontjától,
# 3 láb széles)
3 6 6 3:w
```

4. lista Adatok a tetőszerkesztő számára (roofdata)

```
0 0 p
21 0 p
21 10.5 b
0 10.5 b
0 0 p

# Szegélyek
c:3 0 3 7.5
c:18 0 18 7.5
```

tengely körüli forgatást jelent) és új helyzetbe való eltolásra (`-t x y z` jelenti az x y z távolságokkal való eltolást). A különböző lehetőségek többször is használhatóak az utasításon belül, tetszőleges sorrendben. A műveletek olyan sorrendben hajtódnak végre, ahogy a parancssorban megjelennek. Figyeljünk a jelenetszerkesztő program által használt alapértelmezett helyzetre. A legtöbbjük az origóban egy jelzősarkot helyez el.
Az első ábrán a `genbox` által létrehozott kockákon látható az `xform` hatása. A képen a nézőpont a +z tengelyen helyezkedik el, és az origó felé tekintünk. A piros tengely a +x, a zöld pedig a +y. A kék kocka a `genbox` hívásának eredeti eredményét mutatja. A piros kocka ugyanennek a `genbox` hívásnak az eredménye a következő `xform` műveletek után:

```
!genbox redplastic box1 .5 .5 .5 | xform -rz
45 -t 2 0 0
```

A zöld kocka pedig:

```
!genbox greenplastic box2 .5 .5 .5 | xform -t
2 0 0 -rz 45
```

Az anyagtípusok, mint a ***redplastic*** (piros műanyag), közvetlenül ezek előtt az utasítások előtt lettek meghatározva, de a listákon nem szerepelnek. Látható, hogy a műveletek egymásutánja hogyan befolyásolja a működést, és ezáltal a kimenetet.

Bonyolultabb jelenetek

Számos olyan jelenetszerkesztőt írtam Perlben, ami rönkkabinok és rönkházak összeállítására használható (ezek természetesen elektronikus formában is hozzáférhetők). Ebben a cikkben ezek közül a `genlogwall`, `genlog` és `genroof` nevűt fogom alkalmazni. Kimeneteik mértékegysége a hüvelyk, még akkor is, ha a kényelmesebb használat érdekében a bemenet lábban van megadva. Az általam használt anyagok szintén megtalálhatóak az elektronikus terjesztésben.
A `genlog` segédprogram egy sapkás hengert fog eredményezni, aminek a középpontja a +x tengelyen helyezkedik el (*2. kép*). Ehhez több értékre van szükség:

```
genlog anyagnév hossz_láb átmérő_hüvelyk
```

Az anyagmeghatározásnak már szerepelnie kell a jelenetben, a nevét magunk találjuk ki. Az általam használt előre meghatározott anyagok fájlja az egyes irányoknak megfelelően három faanyaggal rendelkezik: ***xpine***, ***ypine***, ***zpine***. Olyan anyagot kell választanunk, amelyik illik a rönk végső illesztéséhez.
Ha egy tíz láb magas, nyolc hüvelyk átmérőjű, a +z irányba mutató póznát szeretnénk a (15ft; 0ft; 0ft) kezdőponton felállítani, akkor a következő parancsot kell kiadnunk:

```
!genlog zpine mypole 10 8 | xform -ry 90 -t
180 0 0
```

Ne felejtsük el, hogy az `xform` számára megfelelő mértékegységeket kell alkalmaznunk: 180 hüvelyk 15 lábnak felel meg.
A `genlogwall` segédprogram a következő formában használható:

```
genlogwall anyagnév hossz_láb magasság_láb
rönkátmérő_hüvelyk [nyílás_adat_fájl]
```

A `nyílás_adat_fájl` nem kötelező érték, ebben adhatjuk meg, hogy milyen nyílásoknak kell a falban lenniük, és miket kell ezekbe a nyílásokba helyezni (például ajtót vagy ablakot). Most a könnyebbség kedvéért a *3. képen* látható alaprajzról dolgozzunk. Négy fal látható a képen, amelyek mindegyike 15 láb hosszú. A házikó délnyugati sarkát választottam (0;0;0) pontnak, az x értéket kelet felé, az y-t északi irányban növelve, a z növelésével pedig felfelé haladhatunk. Ez a tájolás dél felé fordítja a házat, s `gensky` szabályosan előállított égboltjának megfelelően. A `genlogwall` az előállított falat mindig a (0;0;0) pontból indulva az x tengelyre fekteti, ahogy az a *4. képen* látható.
A nyílások adatfájljának szerkezete is egyszerű, soronként egy nyílást írhatunk le benne:

```
nyílás_alja_láb nyílás_teteje_láb
nyíláskezdet_láb szélesség_láb[:w|d]
```

Az első két érték a padlótól számítandó, az utóbbi kettő a fal indítóélétől (`x=0`, x növekszik). A végén lévő nem kötelező címke azt jelzi, hogy ajtóval (`d`) vagy ablakkal (`w`) szeretnénk-e kitölteni nyílást. Tervem két ilyen nyílásleíró fájl hívását tartalmazza (lásd a *2. és 3. listákat*). Többszörös falakhoz újra felhasználhatjuk ugyanazokat az adatokat, de csak akkor, ha a nyílásokat is ugyanúgy akarjuk megadni.
Házikónk utolsó művelete a tető elkészítése. Egy szokványos tető előállítása nem egyszerű dolog, ezért a `genrooftool` egy kicsivel jobban megdolgoztatja a felhasználót, mint az eddigi műveletek.

A `genroof` a tető sík részeit készíti el; a teljes tetőt a művelet ismétlésével és az `xform` használatával állíthatjuk elő.
A `genroof` működéséhez egy adatfájlra van szükség a tetőrész sarokpontjainak x-y koordinátáival (lábban kifejezve), amiket az alaprajzról az élek mentén az óramutató jártásával ellentétes irányban olvasunk le. A sarokpontok mindegyikének az első (pozitív) síknegyedben kell lennie, a tetőéleknek pedig az x tengellyel párhuzamosan kell futniuk.
A pontok meghatározásához az alaprajzot úgy forgassuk, hogy a tető éle balról jobbra fusson. Most egy picit feledkezzünk el a tető alsó feléről, és a felső rész bal alsó sarkát tekintsük az új kezdőpontunknak. Ekkor a pontjaink: (0;0), (21;0), (21; 10,5), majd ismét (0;0). Eddig rendben.
A pontokat külön sorban kell felvinnünk, a koordinátákat szóközzel elválasztva, és a sor végén jelölve, hogy a pont a tető aljához (`b`), a közepéhez (`mp`) vagy a tetőélhez (`p`) tartozik-e. Ez azért szükséges, mert szabálytalan tetőrészek létrehozására is lehetőségünk nyílik.
Szükség van továbbá az oromszegélyek megadására, ha azt szeretnénk, hogy a `genroof` fával töltse ki ezeket is. Ha a tető tájolása szerint nézünk az alaprajzunkra, látható, hogy a szegélyeknek a fal mentén a (3;0)→(3;7,5) és (18;0)→(18;7,5) koordináták szerint kell haladniuk. Adjuk ezt az információt is a tető adataihoz a `c:` előtaggal. A *4. lista* mutatja a házikóhoz tartozó tető teljes adatfájlját.
A `genroof` indításához tartozó parancssor pedig a következő:

```
genroof -o túlnyúlás_láb típusnév név
tetőadatfájl magasság_láb vastagság_hüvelyk
```

A túlnyúlás értéke a segédprogram számára lehetővé teszi, hogy beállítsa a tetőrész helyzetét, így nyugodtan a fal magasságának megfelelő helyzetbe mozgathatjuk, nem kell a lejtés és a csatlakozás részleteivel foglalkoznunk. Az 5., már teljes házikóhoz tartozó listán látható, hogy a tetőrészek z irányú mozgatásával a tető illeszkedik a fal magasságához, annak ellenére, hogy a túlnyúló rész a fal felső éle alá lóg. Mivel tetőnk szimmetrikus, ugyanazt a `genroof` utasítást egy másik `xform`-mal minden gond nélkül használhatjuk a másik tetőrész létrehozásához.

A jelenet megtekintése

A Radiance része egy `rad` nevű segédprogram, ami a Unixban használt `make` parancshoz hasonlóan működik. A `rad` bemeneti fájljában tárolt változók határozzák meg a jelenet leképezésének módját, és azt, hogy a fény szimulálásához szükséges sok egyéb program mi módon kerüljön meghívásra.
A 6. lista egy megjegyzésekkel ellátott példát mutat erre.
A legtöbb olyan változó, amihez fájlnév tartozik, annyiszor adható meg, ahány fájl a hozzárendeléshez rendelkezésre áll.
A `view` változó szintén többször megadható. Minden egyes `view`-meghatározás egy újabb kép létrehozását eredményezi. Meg kell határoznunk a nézet nevét, a nézőpont helyét (`-vp` nézőpont), az irányt, amerre nézni kívánunk (`-vd` a nézet irányának vektora), és a felfelé mutató irányt (`-vu` felfelé mutató vektor). Kedvelem a `-vt` beállítást is, amivel nagy látószögű (halszemoptikás) nézetet hozhatunk létre.
A különféle minőségi beállításoknál a `H` érték használata nagyon időigényes képelőállítást eredményez (több mint tíz óra egy 2 GHz-es gépen). A legtöbb esetben erre nincs is szükség, az `M` beállítás is remek eredményt ad. Az `L` az interaktív leképezéshez használható. A programleírás és egy kis kísérletezgetés segíthet abban, hogy megtaláljuk a jelenetünkhöz legjobban illő beállítást.

Ha azonnali nézetet kívánunk, az alábbi parancsot használjuk:

```
$ rad -o x11 cabin.rif
```

A kép túl világosnak és fakónak tűnhet ebben a nézetben. Az E billentyű, majd egy ezt követő ENTER megnyomásával egy fényes pontra kattintva a képen javíthatjuk az expozíciós időt. Ezzel nem kell megvárnunk a leképezés befejezését sem.
Az expozícióval annyiszor próbálkozhatunk, ahányszor csak akarunk. A Radiance képadatainak dinamikatartománya messze meghaladja a monitorunkét, ami azt jelenti, hogy anélkül kaphatunk teljesen sötét vagy tiszta fehér képet, hogy veszítenénk a kép adataiból. A képalkotás ezen a téren teljesen eltér a normál képfájloktól, ahol a képet túl fényesre állítva maradandó adatvesztés lehet az eredmény.
Ha az interaktív leképezést használjuk, a ***rif*** fájlból az `L` paranccsal más nézetet is betölthetünk. Ha például a ***rif*** fájlban van egy ***interior*** nevű belső nézetünk, az `L interior` begépelésével betölthetjük a nézetet. Saját kezűleg is felvehetünk egy nézetet a V billentyű és ENTER megnyomásával, csak a kért adatokat kell megadnunk. Az interaktív leképezésből a Q gomb és az ENTER megnyomásával léphetünk ki.
Az összes nézet képének előállítása a következő paranccsal történhet:

```
$ rad cabin.rif
```

Ezt követően a megtekintéshez az alábbi parancsot használhatjuk:

```
$ ximage *.pic
```

Az ximage programban is beállíthatjuk az expozíciót. A képre kattintva és az A billentyűt megnyomva az expozíció önműködően beáll, a H jelenti az emberi szem reakciójának megfelelő beállítást, az = (egyenlőségjellel) pedig arra a képpontra állítjuk, amelyikre rákattintunk az egérrel. Az *5.* és *6. kép* házikónk két nappali képét mutatja.
Az ximage arra is lehetőséget nyújt, hogy egy képterület átlagos fényadatait kiolvassuk. A kívánt tartományra húzzunk egy téglalapot, majd nyomjuk meg az ***L*** gombot a fénysűrűségnek (luminancia) vagy az ENTER-t a fényességértékeknek a leolvasásához.
A számok fizikai jelentéséről a ➲ http://www.intl-light.com/handbook/rad.html címen találhatunk rövid ismertetést. A képből való kilépésre a ***Q*** gombot használhatjuk.
A hely korlátozottsága miatt ebben a cikkben a Radiance lehetőségeinek csak nagyon kis részét mutathattam be. Ha a jeleneteinket a mindennapi élet egyéb tárgyaival tovább szeretnénk bővíteni, mindenképpen látogassunk el a Radiance weboldaláról elérhető honlapokra, ahol érdekes bútorokra, növényekre lelhetünk.

A cikkhez tartozó listák megtalálhatóak a 49. CD Magazin/ Radiance könyvtárában.

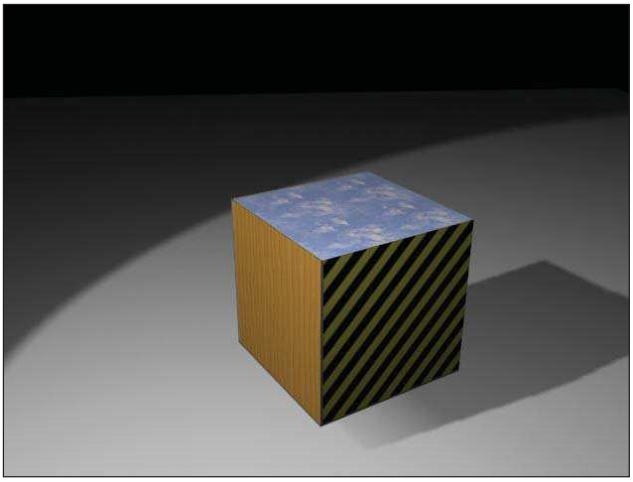
Linux Journal 2003. június, 110. szám

Anthony W. Kay
Programozóként dolgozik az Oregon állambeli Eugene-ben. Ha éppen nem fák és építőanyagok szimulációján dolgozik, túrázni indul, hogy élőben is tanulmányozhassa őket.

Felületi mintázatok a Blenderben (3. rész)

Sorozatunk e részében egy olyan hatékony eljárást szeretnék bemutatni, amivel egy adott tárgyon pontosan a kívánt helyzetben helyezhetők el a mintázatok.

Ennek az eljárásnak a neve angol szakkifejezéssel UV-mapping, amit UV-térképezésre fordíthatnánk le. A módszer alkalmazása során a Blendernek pontosan meg tudjuk adni, hogy a tárgyat alkotó elemek csúcspontjaiban (háromszögek, négyszögek) a mintázatként használt kép melyik pontja jelenjen meg. Ezt az eljárást fogjuk majd felhasználni a Blender játékmotorjának a megismerése során. Játékfejlesztéskor pedig különösen elterjedt megoldás, hogy a háromdimenziós objektumokat kevés alkotóelemből építik fel (ez úgynevezett low-poly modelling, azaz alacsony poligonszámú modellezés), ám annál nagyobb gonddal dolgozzák ki a mintázatokat, ezzel érve el a gyors megjeleníthetőség mellett is szép hatást.
Az egyszerű megoldással kezdeném az ismertetést, első megközelítésben egy kockának minden oldalát különféle mintával borítjuk be. Még mielőtt a Blendert munkára fognánk, készítsük el a mintázatot alkotó képeket. Mivel a tárgyat alkotó elemek helye pontosan ismert, ezt „kézzel" is megrajzolhatjuk. Én az XFig programmal készítettem egy négyzetet, majd még öt példányban lemásoltam, ahogyan az általános iskolai tanulmányaim során a papírkocka hajtogatásakor tettem. Ezt a kiindulási képet PNG formátumba exportáltam, majd a Gimppel kifestettem. Kiválasztottam a megfelelő négyzet belsejét, és mintával töltöttem ki mind a hat négyzetet. A gyakorláshoz el sem kell készíteni őket, ugyanis ezek a képek a CD-mellékleten is megtalálhatók. A képek elkészülte után indítsuk el a Blendert.
Elsőként hozzunk létre egy új kockát a ***Főmenü-Add-Mesh-Cube*** menüpontok kiválasztásával. Miután a kocka megjelent, az A billentyűvel szüntessük meg az összes oldalának kijelölt állapotát, és a TAB-bal kapcsoljunk át tárgyszerkesztő módba. Az UV-térképek elkészítéséhez át kell váltanunk ***Vertex-paint*** módba. Az alsó panelen látható egy rádióaktivitást jelző kis ikon. Nem kell megijednünk tőle, viszont a váltáshoz a tőle balra lévő gombot kell használnunk, ami leginkább egy kis ecsethez hasonlítható. A következő lépésben – a már ismert módon – a szerkesztőnézetet osszuk kétfelé, és a jobb oldali nézetben nyomjuk meg a SHIFT-F10 billentyűket. Ezzel a nézetet átváltottuk az UV-szerkesztő módba, és a ***Load*** gomb alkalmazásával betölthetjük az előre elkészített képet. Megjegyzendő, hogy a Blender csak a PNG, a JPG és a TGA formátumú képeket jeleníti meg, és ezeket a formátumokat képes felhasználni a szerkesztőben is. Tehát a kép betöltése után elöl-, oldal- vagy felülnézetben ki kell választanunk azokat a háromszögeket, amelyeken majd a mintázatot látni szeretnénk.
Ezt a legkönnyebben úgy tehetjük meg, hogy átkapcsolunk pontszerkesztő módba, és a nézet csekély elforgatása után kiválasztjuk a megfelelő csúcspontokat. Ezután váltsunk vissza az előző nézőpontba, és nyomjuk meg ismételten a TAB gombot, hogy a Blenderrel elkészíttethessük a kijelölt háromszögek kiterített képét. Kijelölt kockánkon használjuk a U billentyűt, és a megjelenő menüből válasszuk ki a ***Cube***, a ***Standard1/1*** vagy a ***Window → From Window*** menüpontot. Bármelyik mellett döntünk is, a továbbiakban az elkészített hálóval a jobb oldali nézetben kell majd dolgoznunk. Ebben az egyszerű esetben a háló csupán egy négyzetre egyszerűsödik, hiszen a kockának az ebből a látószögből alkotott képe egy négyzet lesz. Bonyolultabb formák esetén az elkészített háló úgy képzelhető el, mintha a kamera lencséje sík lenne, és erre terítené ki a Blender a kijelölt háromszögeket. Tehát a feladat annyi, hogy a Blender által kiterített háló pontjait a mintázatként felhasználni kívánt kép megfelelő pontjaira kell igazítani. Ebben a példában a képen a négyzet csúcsait azokra a pontokra igazítsuk, ahol a rajzolás során az oldalak határait megrajzoltuk. Ezek után a tárgyat ábrázoló nézetben kapcsoljuk ki a háromszögszerkesztő módot az F billentyű használatával. Mivel a kocka minden oldalára különféle mintázatot terveztünk, a fenti lépéseket a fennmaradó öt oldalra is el kell végezni, hogy az eredmény megfeleljen a tervben foglaltaknak.



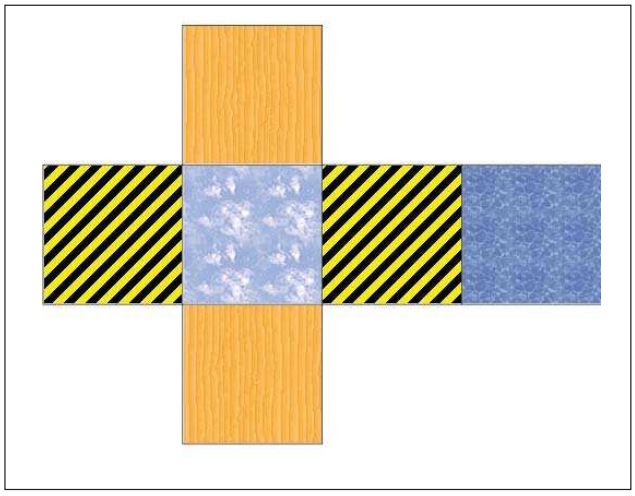
1. kép Munkánk eredménye

Ennyi munka után végül tekintsük meg az eredményt! Adjunk fényforrásokat a jelenethez, majd számoltassuk ki a képet. Láthatjuk, hogy a kockánk teljesen szürke. Adtunk-e neki valamikor anyagot? Eddig még nem, tehát itt az ideje ennek is. Kapcsoljunk az anyagszerkesztő nézetbe, a kockához pedig rendeljük hozzá az alapértelmezett anyagot. Ha most a képet újra kiszámítanánk, semmi változást nem láthatnánk, hiszen amíg semmilyen anyagot nem rendeltünk a tárgyhoz, addig az alapértelmezett tulajdonságokkal bír. Ahhoz, hogy a Blender figyelembe vegye a beállított koordinátákat, kapcsoljuk be az anyag színének meghatározása alatt a ***TexFace*** kapcsolót. Így a beállított mintázat már a számítások során is megjelenik a tárgyon, ahogyan ez *1. képünkön* is látható.
Hogyan tudjuk a mintázatokat könnyedén létrehozni? Kapcsoljuk az egyik nézetet az UV-térképező módba, míg a másikat a 3D-megjelenítés egy tengelymenti nézetére (felül-, oldal- vagy elölnézet). A F billentyűvel váltsunk UV módba és jelöljük ki azokat a háromszögeket, amelyek közel

Az uvexport.py

```
import Blender210

print "--------------------------"
print "  UV coordinate exporter  "
print "      for Blender 2.23    "
print "  created by Zooli        "
print "e-mail: dzooli@freemail.hu"
print "version: v0.9             "
print "--------------------------"
scene=Blender210.getCurrentScene()
for o in scene.objects:
    ob210=Blender210.getObject(o)
    if Blender210.isMesh(obj):
      print "Object: "+obj
      m210=Blender210.getMesh(o))
      print "  Filename: "+o+"_tex.uv"
      f1=open(o+"_tex.uv","w")
      for c in m210.texcoords:
          tc = []
          tc.append(c)
          f1.write("%s %s\n" % (tc[0]))
      f1.close()
      print "  export ok."
print "All coordinates exported."
```



2. kép A kiterített kocka

merőlegesek a nézési irányra. Az F billentyűt használjuk ismételten, majd az U billentyű hatására megjelenő menüből válasszuk ki a *„From window”* menüpontot. Ezek után a térképező nézetben megjelennek azok a háromszögek, amelyeket kiválasztottunk. Mozgassuk ezeket olyan pozícióba, hogy a későbbiekben a többi háromszögünknek is maradjon hely. Ezt a folyamatot az adott nézőpontból folytassuk addig, amíg minden közel merőleges háromszögnek meghatároztuk a mintázatpozícióit. Majd váltsunk nézetet és ismételjük meg a többi háromszöggel is ezt a folyamatot. Előfordulhat, hogy egy képre nem fér rá minden háromszögünk, ilyen esetekben újabb képet is megadhatunk az UV-térképező nézetben. Ezután jelöljük ki azokat a háromszögeket, amelyeknek már meghatároztuk a koordinátáit, és nagyítsuk ki az UV-térképező nézetet az alsó fejlécén található négy részre osztott gomb használatával. Ez balról a második gomb. A „szürke +” billentyűvel nagyítsuk a térképet olyan méretűre, hogy a képernyő legnagyobb részét betöltve még minden rész látható legyen. Készítsünk képernyőmentést, majd egy rajzoló programmal vágjuk ki a szükséges részt és rajzoljuk meg a mintázatokat. Amikor a mintázatokat rajzoljuk, a kiszínezett részekkel fedjük le teljesen a Blender által készített szaggatott vonalakat, és az sem baj, ha ezeken egy kissé túlfestünk. Amikor a mintázatot újra betöltöttük a Blenderbe a csúcspontok mozgatásával pontosítsuk a háromszögek helyzetét az UV-térképező nézetben. A pontosság megállapításához nem kell mindig újra kiszámítanunk a képet, mert a ALT-Z billentyűkombinációval a szerkesztő nézetben is megjeleníthetjük a mintázatot. Ezzel a módszerrel készült mintázatok a CD-mellékleten is találhatók.

Itt jegyezném meg, hogy az előbb ismertetett módszerrel egy – a játékfejlesztés során használt – segédeszközt is megtakaríthatunk, ugyanis ilyen koordinátákat az elkészülő modellekhez nem kell külön segédprogrammal létrehozni, ezt a modellező-programon belül is megtehetjük. Játékfejlesztés során előfordulhat, hogy ezekre a koordinátákra később is szükségünk lesz, mentésüket azonban a Blender nem támogatja. Szerencsére lehetőségünk nyílik Python programokkal kiegészíteni a Blender lehetőségeit, ahogyan azt a mellékelt programlistán is láthatjuk. Ezt a programot célszerű a Blender 2.23 változatával használni, ugyanis a későbbi változatokban nem működik, mert azokban a Python-bővítményekben használható eljárások folyamatosan változnak, így a program már az első sort sem tudja végrehajtani a ***Blender210*** függvénykönyvtár hiánya miatt (lásd *listánkat*).

Ez az egyszerű kis kiegészítés lehetővé teszi, hogy a Blenderben létrehozott és beállított koordinátákat a továbbiakban felhasználjuk például a saját játékunkban a mintázatok elhelyezéséhez. A könnyebb használhatóság érdekében a CD-mellékleten megtalálható a program egy kissé átalakított változata, és egy jelenet, amiben csak ez a program szerepel, tehát akár azt átalakítva vagy abból a segédprogramot betöltve bárki tetszése szerint tovább alakíthatja és használhatja. Ezután még szükség lesz magára a modellre is. A Blenderben a SHIFT-F2 billentyűkkel DXF formátumban menthetjük a modelleket, aminél egyszerűbbet keresni sem kell. Leendő játékunkban akár közvetlenül, akár további átalakítások után könnyedén felhasználhatjuk az így létrehozott tárgyakat.

További fontos tudnivaló, hogy eddigi tapasztalataim szerint a Blender játékmotorja csak az így elkészített UV-térképekkel ellátott tárgyakon jeleníti meg a mintázatokat, tehát tárgyalásakor ennek az eljárásnak különösen fontos szerepe lesz. Az alábbiakban rövid ízelítőt adok a Blender képességeiből a részecskerendszerek megjelenítésének területén. A részecskerendszerekről elegendő annyit tudnunk, hogy a szimuláció során minden részecske mozgását a program egyenként számítja ki, és a rendszer kialakításában résztvevő részecskék egymással is kölcsönhatásban állnak. Ezt a témát sajnos nem lehet teljességében bemutatni, hiszen nagyon sok múlik a személyes alkotókészségen, így itt egyetlen példán keresztül csak az alapokat tudom bemutatni.

Első lépésként hozzunk létre egy kúpot a ***Főmenü/Add/Mesh/Cone*** menüpontokat kiválasztva. A Blender alaphelyzetben a részecskék kiindulási irányát a tárgy háromszögeit alkotó pontok normálvektorával megegyező irányként állítja be. Ebben a példában egy szökőkúthoz hasonló hatás megalkotását szeretném bemutatni, tehát a vízcseppeknek kezdetben felfelé kell halad-

Shift-F10	UV-szerkesztő mód
Shift-F2	Mentés DXF formátumban
F	Háromszög-kiválasztó mód
F10	Számítás beállításai

niuk, ezt követően lefelé. Ha a részecskéket a kúpot változatlanul hagyva állítanánk be, akkor azok lefelé indulnának, másrészt pedig a kúp csúcsából is elszabadulna néha egy-egy részecske. Ezt elkerülendő töröljük ki a kúp csúcspontját alkotó pontot. Jelöljük ki a csúcsot, majd az X billentyű hatására előkerülő menüben válasszuk ki a ***Vertices*** pontot. Így már nem adunk lehetőséget kósza részecskék felbukkanására. A másik megoldandó feladat a kiindulási irány megváltoztatása. Jelöljük ki a megmaradt pontokat, és oldal- vagy elölnézetben forgassuk el őket 180 fokkal. Ehhez használjuk az R billentyűt, és a pontos forgatás megvalósításához a Ctrl gomb nyomva tartásával forgassuk el a pontokat. Így a részecskék már a megfelelő irányba indulnak. Ezután hozzunk létre egy világoskék színű anyagot, kapcsoljuk be a ***Halo*** tulajdonságot, és az ***Alpha*** értéket vegyük alacsonyabbra. A szebb látvány érdekében kapcsoljuk be még a ***Rings*** kapcsolót is. Ezek a beállítások megfelelnek a célomnak, de kedves olvasóim természetesen szabadon kísérletezhetnek más értékekkel is.

Ami igazán fontos, az magának a részecskerendszernek a beállítása. Az F7 billentyűvel aktiválhatjuk az animációs beállításokat. A nézet közepén egy ***New Effect*** feliratú gombot találunk, amit most használnunk kell. A tőle jobbra lévő listából válasszuk ki a ***Particles*** beállítást. Ennek hatására rengeteg beállítási lehetőség tűnik fel e két gomb alatt.

Kezdjük az elején: a `Tot` értékkel a rendszerben szereplő részecskék legnagyobb számát adhatjuk meg. A `Sta`, az `End` értékek határozzák meg a részecskék mozgásának kezdő és befejező idejét, az animáció képkockáit használva mértékegységül. Ezt a két értéket állítsuk 1-re és 100-ra. A mellettük található ***Life*** értékkel szabályozhatjuk, hogy a részecskék első nemzedéke mennyi ideig szerepeljen a rendszerben, míg a `Keys` jelzi a Blender számára, hogy hány kulcskockát számítson a teljes életciklusuk során.

A következő sorban a `CurMul` azt mutatja, hogy hányadik nemzedék adataira vonatkoznak a mellette lévő értékek.

A nemzedékekről annyit kell tudnunk, hogy amikor egy részecske az életciklusa végére ér, képes újabb részecskéket létrehozni, amiket a Blender szóhasználatában gyermekeknek nevezünk. A valóságban ilyen „gyermekeket” például tűzijáték szemlélése során láthatunk. E beállítás mellett található a gyermekek által használt anyag sorszáma, a `Mult` értékkel pedig beállíthatjuk megszülető „gyermekrészecskék” arányát. Amennyiben ez az érték 0, úgy a részecske az életciklusát befejezve eltűnik, míg a legnagyobb 1-es érték azt jelenti, hogy mindig a `Child` érték (a gombsor utolsó eleme) által meghatározott számú „gyermek” születik. Az előbbi két beállítható érték között találjuk az új nemzedék életének a hosszúságát. Természetesen az új részecskéknek is lehetnek majd utódaik, ezek tulajdonságait úgy határozhatjuk meg, ha a sor elején található `CurMul` értéket növeljük, és a fentiek alapján ismételten beállítjuk a tulajdonságokat. Alább még érdekes a `Randlife` és a `Seed`, amelyekkel az életciklus hosszúságának adhatunk véletlenszerű változást. A következő sorban látható a `Norm` érték, amivel azt határozzuk meg, hogy a tárgyat alkotó háromszögek normálvektorai mekkora hatással legyenek a részecskék kiindulási sebességére, míg az `Ob` értékkel a Blender a tárgy helyzete alapján számítja ki a kezdősebességet. Itt a `Norm` értéket állítsuk 0,9-re, a `Rand` értéket pedig 0,2-re. A `Rand` változóval a részecskék kezdősebességét véletlenszerűen változtathatjuk. A `Tex` beállításával a tárgy felületi mintázata fogja befolyásolni a sebességértékeket. Ez alatt az érték alatt állíthatjuk be, hogy a mintázat milyen módon befolyásolja a kezdősebességeket. A három koordinátatengely mentén külön-külön beállítható, hogy a mintázat mekkora hatással legyen a részecskék sebességére, erre szolgál a `X`, `Y` és `Z` érték. Mellette látható a `Int` kapcsoló, amivel azt adjuk a Blender tudtára, hogy a mintázat képpontjainak világosságértékét vegye figyelembe. A ***RGB*** kapcsoló használatával a mintázat színösszetevői szorzótényezőként szerepelnek majd a kezdősebesség meghatározása során, a ***Grad*** kapcsolóval pedig a világosságértékek változása befolyásolja a részecskék kezdősebességét.

Ezektől a beállítószervektől balra lévő értékekkel a megfelelő koordinátatengelyekkel párhuzamosan állandó értékű erőhatást adhatunk a részecskerendszerhez, azaz egy térbeli vektor összetevőit határozhatjuk meg. Itt állítsunk be `(-1)`-es értéket a Z tengely irányában, ami majd a gravitáció szimulálására lesz alkalmas.

Adjunk a jelenethez fényforrást, és a kamerát helyezzük el nagy távolságra a részecskerendszertől. Nagy távolság alatt azt kell érteni, hogy a látvány teljességének élvezetéhez a részecskéket kibocsátó tárgynak célszerű a kamera látóterén belül lennie. Később a kamerát természetesen a megfelelő helyre tehetjük, de kezdetben az egész részecskerendszert ajánlatos láthatóvá tenni.

Ezek után már csak a végeredmény kiszámítása maradt hátra. Az F10 billentyűvel váltsunk át a számítási beállításokat megjelenítő nézetre, és a ***Anim*** gomb oszlopának alján található `End` értéket állítsuk százra. A nézet bal szélén még találunk egy kapcsolót ***DispView*** felirattal. Ezt bekapcsolva a számítási eredmények közvetlenül a kamera látóterét megjelenítő nézetben lesznek láthatók. Kapcsoljunk a kamera nézetére, és kattintsunk a ***Anim*** gombra. Ekkor a szemünk előtt fog kibontakozni a végeredmény. Természetesen ezt akár rögtön a háttértárra is menthetjük, méghozzá úgy, hogy az alsó nézet bal oldalán a ***pics*** mezőbe beírjuk az állomány nevét, és az ***Anim*** gombtól jobbra, az az alatt található, alapértelmezésben ***Targa*** feliratú listáról kiválasztjuk valamelyik ***AVI*** típust. A CD-mellékleten található animáció elkészítése során az ***AVI JPEG*** típust használtam, ezért ebben az esetben is ezt tudom javasolni. Mindezeket a beállításokat célszerű az animáció kiszámítása előtt elvégezni, ellenkező esetben a képsorozatot újra ki kell számíttatni a programmal.

Ebben a hónapban ennyi újdonság és lehetőség bemutatása véleményem szerint elegendő alapot adhat arra, hogy továbbra is mindenki kedvét lelje a program használatában. Az új billentyűkombinációk szokás szerint az összefoglaló *táblázatban* találhatók meg.

Kellemes alkotást kívánva búcsúzom.

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.

Játékprogramozás az SDL programkönyvtárral

Saját játékainkhoz is felhasználhatjuk a Tux Racer és a Civilization mögött megbújó programkönyvtárat.

Az SDL (Simple DirectMedia Layer, ➲ http://www.libsdl.org) egyszerű, de hatékony felületfüggetlen játék- és multimédia-fejlesztő programkönyvtár, amelyet ***Sam Latinga*** fejlesztett ki, amikor a Loki Software játékkészítő cégnél dolgozott. A Loki az SDL-t használta kereskedelmi játékainak fejlesztésekor. Az SDL-t a több operációs rendszerrel dolgozó játékkészítők igényeit szem előtt tartva fejlesztették ki, és többek között a következő játékok linuxos változatához használták: Maelstrom, Hopkins FBI, Civilization: Call to Power, Descent 2, MythII: Soulblighter, Railroad Tycoon II és Tux Racer. Az SDL honlapján több száz SDL-t használó játék és alkalmazás van felsorolva.
Az SDL hivatalosan támogatja a Linux, Windows, BeOS, Mac OS, Mac OS X, FreeBSD, OpenBSD, BSD/OS, Solaris és Irix operációs rendszereket. Működik Windows CE, AmigaOS, Atari, QNX, NetBSD, AIX, Tru64 Unix és SymbianOS környezetben is, de ezek az operációs rendszerek hivatalosan még nem támogatottak. Ebből következően az SDL segítségével megírt alkalmazás csekély erőfeszítéssel az összes felsorolt operációs rendszer alá átvihető. Az SDL a hordozható játék- és multimédiás alkalmazások fejlesztését minden ma használatos nagy operációs rendszerre támogatja.

Az SDL telepítése

A nem túl régi Linux-terjesztéseknek eleve része a teljes SDL. Például a saját Red Hat 8.0 rendszeremen nyolc programot is találtam a */usr/bin* könyvtárban, amelyek az SDL-től függtek. A következő parancsok segítségével megállapíthatjuk, hogy az SDL programkönyvtárak és a C/C++-fejlécállományok telepítve vannak-e a rendszerünkön:

```
locate SDL.h
locate libSDL
locate sdl-config
```

Ha a parancsok mindegyike azt jelzi, hogy az állomány megtalálható, akkor nagy valószínűséggel teljes SDL-telepítéssel bírunk, és csak arról kell meggyőződnünk, hogy elég friss-e. Az ***sdl-config*** program visszaadja az SDL változatszámát és a fordítóprogramnak megadandó kapcsolókat. Ha az ***sdl-config*** program telepítve van, adjuk ki az `sdl-config --version` parancsot, hogy megtudjuk a telepített SDL változatszámát. Ha az eredményül kapott változatszám 1.2.4-nél kisebb, telepítsük a programkönyvtár újabb változatát. A legtöbb nyílt forrású projekthez hasonlóan az SDL is folyamatos fejlesztés alatt áll, így ha felhasználjuk a saját fejlesztéseinkhez, érdemes rendszeresen figyelni az új változatok megjelenését, vagy az SDL-es levelezőlisták egyikéhez csatlakozva követni a frissítéseket.
Ha az SDL nincs telepítve, töltsük le és telepítsük. A terjesztések általában tartalmazzák az előre lefordított SDL-csomagokat, ezért a keresést itt kezdjük. Ha a csomagok kellően frissek, az a legegyszerűbb, ha a terjesztésünkhöz való ***dev*** vagy ***devel*** végződésű SDL-csomagokat telepítjük.
A cikkben bemutatott forrás mellett megtalálható az `sdl-install.sh` parancsállomány, ami letölti és telepíti az SDL 1.2.5-ös változatát és az összes bővítmény-programkönyvtárat. A parancsállományt rendszergazdaként kell futtatni abban a könyvtárban, amibe az SDL forrását tartani szeretnénk. Ha nem az `sdl-install.sh` parancsállományt használjuk, akkor a megadott weboldalakról le kell töltenünk a fájlokat, és kicsomagolás után a ***README*** állományokban leírt módon telepítenünk kell őket. Az új telepítést a következő paranccsal ellenőrizhetjük:

```
sdl-config --version
```

Ha a parancs nem fut le, vagy az éppen telepített változatnál régebbit jelez, akkor a telepítés sikertelen. Tapasztalataim szerint ez akkor következik be, ha nem követjük az utasításokat, vagy egy régi SDL-változat fenn van egy másik helyen már . Ha az ***sdl-config*** egynél több helyet sorol fel, akkor vagy töröljük a régi SDL-telepítést, vagy az újat telepítsük a régi helyére. Az `sdl-install.sh` állomány bemutatja, hogy a `./configure --prefix` használatával az SDL tetszőleges helyre telepíthető, de a legkönnyebb és legbiztonságosabb az alapértelmezett hely használata.
Az SDL leírása a ➲ http://www.libsdl.org/docs.php címen található meg. Az online leírás helye pedig az ➲ http://sdldoc.csn.ul.ie. A támogató programkönyvtárak leírása vagy a letöltési oldalukról érhető el, vagy a forráskódhoz mellékelik, esetleg a ***.h*** állományokba van beágyazva. Az SDL-hez példaprogramok is tartoznak, és a támogató programkönyvtárak jól használhatók a saját fejlesztéseink elindításához.

SDL-példaprogram

A ***bounce.cpp*** (elérhető az ➲ ftp.ssc.com/pub/lj/listings/issue110/6410.tgz címen, illetve a 49. CD Magazin/SDL könyvtárában) állomány egy játékprogram forráskódját tartalmazza. A bevitelt és a grafikát az SDL, a TrueType betűkészletek betöltését az ***SDL_ttf*** programkönyvtár intézi. Maga a játék egy kicsit több mint 1300 sor C++-kód. A csomag tartalmazza a forráskódot, a képeket, a TrueType betűkészletet, a `Makefile`-t, az `sdl-install-sh` állományt, valamint a játékban felhasznált betűkészlethez és a képekhez tartozó felhasználói szerződést. Több ideig eltarthat a játékhoz jogtisztán felhasználható betűkészleteket, képeket és hangokat találni, mint magát a játékot megírni.
Kezdjük meg az SDL tanulmányozását a Bounce játék forrásának letöltésével és kicsomagolásával (`tar -xzvf bounce.tar.gz`). Ezután futtassuk a `make` parancsot. A lefordított programot a `bounce` parancs kiadásával indíthatjuk.
A `bounce -fullscreen` parancs hatására a program teljes képernyős üzemmódban indul. A játék lényege, hogy a Föld kóborol a Naprendszerben, és fennáll a veszély, hogy a Napba zuhan. Feladatunk, hogy a Földet távol tartsuk a Naptól, amit úgy érhetünk el, hogy nekiütközünk a Holddal. Minden alkalommal pontot kapunk, ha a Földet eltaláljuk a Holddal, és a játék magának könyvel el egy pontot, ha a Föld összeütközik a Nappal. A játék célja az SDL képességeinek a bemutatása, nem pedig a világ legérdekesebb játékának a megalkotása.

Az SDL indítása

Az SDL-t el kell indítani, mielőtt használhatnák a képességeit. Erre szolgál az `SDL_Init()` függvény:

```
if (-1 == SDL_Init((SDL_INIT_VIDEO |
                    SDL_INIT_TIMER |
                    SDL_INIT_EVENTTHREAD)))
{
  ...
}
```

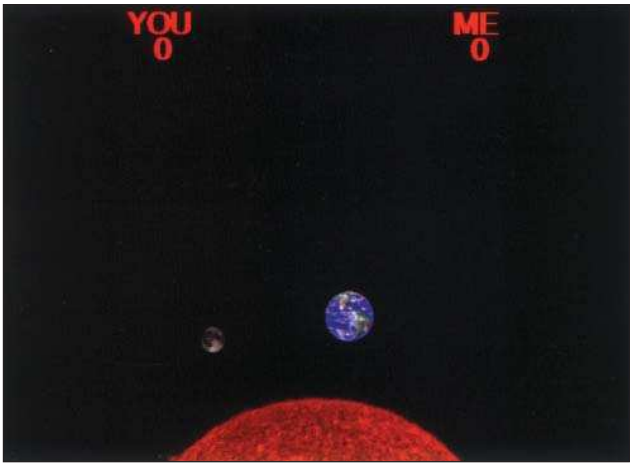
Az `SDL_Init()` függvénynek átadott érték adja meg az alrendszert, amit el kell indítani. Itt a képmegjelenítést, az időzítést és a szálakon alapuló eseménykezelést kapcsoltuk be. Használhattuk volna az `SDL_INIT_EVERYTHING` értéket is, ami az összes SDL-részt bekapcsolja, de célszerű csak azokat a részeket elindítani, amiket ténylegesen használunk a programban. Nincs értelme elindítani a botkormányt vagy a CD-ROM-ot kezelő részt, ha úgysem használjuk. Bármikor elindíthatjuk és leállíthatjuk az SDL egyes alrendszereit az `SDL_InitSubSystem()` és a `SDL_QuitSubSystem()` függvényekkel.
Fontos, hogy az SDL-t hívással leállítsuk az `SDL_Quit()`, még mielőtt a programunk leáll. Az `SDL_Quit()` az összes SDL-alrendszert leállítja, felszabadítja az SDL által használt rendszererőforrásokat, és helyreállítja a videomódot. Jó gyakorlat az `atexit()` használata, hogy biztosak legyünk benne, hogy az `SDL_Quit()` lefut a programunk befejeződésekor. Ha elmulasztjuk meghívni az `SDL_Quit()` függvényt, a számítógép furcsa videomódban maradhat.

A videomód beállítása

A videomód kiválasztásakor el kell döntenünk, hogy ablakban vagy a teljes képernyőn szeretnénk-e futtatni az alkalmazást. Azután meg kell adnunk az ablak vagy a képernyő méretét. Ha az ablak mellett döntünk, meg kell adnunk, hogy a felhasználó átméretezheti-e. Ezután ki kell választanunk a képernyő színmélységéhez való alkalmazkodás módját. A Bounce programban valami ilyesmit használtam:

```
options = SDL_ANYFORMAT |  SDL_FULLSCREEN;
screen = SDL_SetVideoMode(640, 480, 0, options);
```

Az első két érték a program ablakának vagy képernyőjének a szélességét és a magasságát adja meg képpontokban. Az adott szélességet és magasságot csak akkor használhatjuk teljes képernyős módban, ha az ***XF86Config-4*** állomány (bizonyos X-változatokban XF86Config) „screen” fejezetében szerepel a megadott méret. Ha a Bounce nem fut teljes képernyős üzemmódban a gépünkön, akkor a 640×480-as képernyőfelbontás valószínűleg nincs beállítva az ***XF86Config-4*** állományunkban. A harmadik érték adja meg a képpontokat leíró bitek számát. Ha nullára (0) állítjuk, akkor az SDL a pillanatnyi színmélységet használja. Jobban járunk, ha a játék a pillanatnyi színmélységhez alkalmazkodik, mert ekkor nem kell minden egyes géptípusról, amin a játékunk futhat, számon tartanunk, hogy támogatja-e a kívánt képpontformátumot.
Az utolsó érték segítségével részletes utasításokat adhatunk az SDL-nek a videomód beállításával kapcsolatban. Közel egy tucat lehetőség közül választhatunk. A Bounce programban az `SDL_ANYFORMAT` lehetőséget választottam, ami az SDL-t a lehető legjobb mód választására utasítja. Ez a lehetőség arra kényszeríti, hogy a kód minden lehetséges színmélységhez alkalmazkodjon, de egy kis többletkódolás árán még jobb



A Bounce játék

teljesítményt nyújt. Az `SDL_FULLSCREEN` lehetőség az SDL-t teljes képernyős üzemmódba kapcsolja.
Az `SDL_SetVideoMode()` által visszaadott érték mutató egy `SDL_Surface` szerkezetre. Ez a szerkezet írja le a képernyőt teljes részletességgel. De a `NULL` érték visszakapása nem azt jelenti, hogy mindent megkaptunk, amit akartunk. Ellenőrizzük a szerkezet jelzőbitmezőjét, hogy a kért lehetőségek működnek-e. Úgy tapasztaltam, hogy érdemesebb keveset kérni, és azzal dolgozni, ami van, mert így elkerülhető, hogy operációs rendszertől függő kódot építsünk a programba.
A videomód beállítása után az ablakcím és az ikonnév beállítására használjuk az `SDL_WM_SetCaption()` függvényt. Ez nem kötelező, de a program használata egy kicsit könnyebb lesz tőle:
`SDL_WM_SetCaption("Bounce", "Bounce")`

Az erőforrások betöltése

Mielőtt a Bounce elindulhatna, be kell töltenie és kezdeti értékekkel kell ellátnia a használt erőforrásokat. A Bounce programnak be kell állítania a színeket, be kell töltenie egy pár képet és a szövegek megjelenítésére használt betűkészletet. Mivel a videomódot az `SDL_ANYFORMAT`-ra állítottuk, az összes erőforrást át kell alakítani olyanra, hogy tetszőleges képernyőformátumhoz megfeleljen. A következő két kódsor egy vörös képpontot hoz létre a szükséges formátumban:

```
SDL_PixelFormat *pf = screen->format;

int red = SDL_MapRGB(pf, 0xff, 0x00, 0x00);
```

Az `SDL_PixelFormat` szerkezet a képernyő képpontjainak leírása, és az `SDL_MapRGB()` a képpontot a szabványos 24-bites RGB színformátumból egy olyan képpontértékre alakítja át, ami a kívánt színt jeleníti meg a képernyőn.
A képek betöltése egy kissé bonyolultabb:

```
SDL_Surface *s0, *s1;

s0 = SDL_LoadBMP(name);
s1 = SDL_DisplayFormat(s0);
SDL_SetColorKey(s1,
                (SDL_SRCCOLORKEY |
                 ➥SDL_RLEACCEL),
                black);
SDL_FreeSurface(s0);
```

Az SDL magja tartalmazza az `SDL_LoadBMP()` függvényt, ami egy ***.bmp*** formátumú képet tölt be egy `SDL_Surface` szerkezetbe. Az `SDL_image` sok más képformátum betöltéséhez kínál függvényeket. A kép abban a formátumban lesz, amelyben létrehozták. A megjelenítési formátumra az `SDL_Display_Format()` függvény segítségével alakíthatjuk át. Az `SDL_SetColorKey()` arra szolgál, hogy az SDL-lel közöljük, a kép másolásakor ne vegye figyelembe a fekete képpontokat. Amikor a Föld képét mozgatjuk a képernyőn, a fekete képpontokat nem másoljuk, csak a kerek Föld belső képpontjai érintettek. Az `SDL_RLEACCEL` jelzőbit arra utasítja az SDL-t, hogy az RLE (run length encode) módszer szerint tömörítse a képet. Az RLE-vel kódolt képek másolása gyorsabb.
A Bounce egyetlen TrueType betűkészletet használ, de három különböző méretben és három különböző stílusban. Az ***SDL_ttf*** programkönyvtárat használva írtam egy függvényt, ami betölti a TrueType betűkészletet, 0-tól 127-ig minden ASCII-karaktert megjelenít egy `SDL_Surface` szerkezetben, minden karaktert átalakít a képernyőhöz, és menti a betű magasságát és szélességét, hogy a szövegek kirajzolhatók legyenek a képernyőre.

A főhurok

Az SDL eseményvezérelt beviteli rendszert használ, hasonlóan az X-hez, a Mac OS-hez vagy a Windowshoz. Amikor lenyomunk egy billentyűt vagy megmozdítjuk az egeret, egy esemény kerül a sorba. A program vagy vár az eseményekre az `SDL_WaitEven()` segítségével, vagy lekérdezheti, hogy történt-e esemény az `SDL_PollEvent()` használatával. A főhurok feladata az események feldolgozása, a játék állapotának frissítése, a következő képkocka kirajzolása, majd az egészet elölről kell kezdenie, amíg nem végez.
Az, hogy az eseményekre várunk vagy lekérdezzük őket, a játék egész szerkezetét befolyásolja. Én a várakozást választottam – ekkor a műveleteket szívverésszerű időzítés vezérli. Azért szeretem ezt a megoldást, mert a program akkor dolgozza fel az eseményeket, amikor történnek, és közben a processzorhasználat is kézben tartható. Mindkét jellemző igen fontos a hálózati játékokban.
Az időzítőt a következő kódsorral hozzuk létre:

```
timer = SDL_AddTimer(10, timerCallback, NULL);
```

Ezután az SDL a `timerCallBack` függvényt tízezred-másodpercenként meghívja. Az időzítő visszahívó függvénye esetünkben az `SDL_PushEvent()` függvényt használva küld egy eseményt. Mivel az időzítő visszahívó függvénye külön szálban fut, akkor is tud eseményt küldeni, ha a játék eseményre várva áll. Az időzítő eseményét megkapva a Bounce ellenőrzi, hogy egy újabb képkockát kell-e kirajzolnia. Az időzítő biztosítja, hogy a program nem próbál meg másodpercenként száznál több képkockát kirajzolni, miközben lehetővé teszi, hogy a játék kisebb frissítési sebességgel is működhessen. A gépemen 85 képkocka/másodperc sebességgel fut, ami megfelel a monitorom képfrissítési gyakoriságának.
A Bounce több lapból áll. A főhurok kezeli azokat az eseményeket, amik minden lapnál közösek, például a kilépés az Esc billentyű vagy a szüneteltetés az F1 billentyű hatására. Miután a főhurok megvizsgálta az eseményt, továbbítja a pillanatnyi lapnak. Minden lap egy függvény, ami egy `SDL_Event` típusú értéket vár. Minden lap felelős az események kezeléséért, az idő nyilvántartásáért és a képernyő kirajzolásáért. Bár ez a megközelítés többször ismétlődő kódot eredményez, a programozónak nagyobb a szabadsága és objektumközpontú tervezést eredményez, amiben minden lap a laposztály egy példánya. Az 1. listában (49. CD Magazin/SDL könyvtár) látható példa a főhurok egyes részeit mutatja be, és jól megfigyelhető, hogy az eseményeket hogyan adjuk át az egyes lapoknak.
A `currentPage` teljes hatókörű változó a pillanatnyi lap megvalósítására mutat. Amikor egy lap új lapot akar indítani, létrehozza azt, és a mutatót arra a lapra állítja. A Bounce három lappal rendelkezik, az első lap az üdvözlőképernyő, ami a program indulásakor jelenik meg, a második lap magát a játékot kezeli, és a harmadik a „Nyertél/Vesztettél” üzenetet jeleníti meg.
Az üdvözlőképernyő eseménykezelője így fest (2. lista, 49. CD Magazin/SDL könyvtár).
Amikor ez a kód megkapja az időzítő eseményt, ellenőrzi, hogy mikor frissítette utoljára a képernyőt, és meghívja a `drawWelcome()` függvényt, ami animálja a képernyőt. Ha észleli az egérgomb lenyomását, az `initBounce()` meghívásával átvált a játék lapjára, és átállítja a `currentPage` változót, hogy az a játék lapjára mutasson. A következő alkalommal a főhurok `bounce()` függvénye fog meghívódni.

Animáció

Az objektumok mozgatása az úgynevezett piszkos képpontok módszerével történik, azaz minden képkockából mindig csak egy kis részt rajzolunk újra. Az objektum régi és új helyzetét egyaránt megjegyezzük. Amikor a Bounce a Földet rajzolja ki, először letörli a piszkos képpontokat, úgy, hogy háttérszínnek tölti fel azokat, ezután a Földet az új helyzetben rajzolja ki. Téglalapok kitöltésére és képek rajzolására ezeket használjuk:

```
SDL_FillRect(képernyő, téglalap, szín);
SDL_BlitSurface(kép, NULL,  képernyő ,
téglalap );
```

Az `SDL_FillRect()` egy `SDL_Surface` szerkezetben (például a képernyőn) egy téglalapot az adott színnel fest ki. A téglalapot az `SDL_Rect` szerkezet határozza meg, a színt az `SDL_MapRGB()` függvénnyel állítjuk elő.
Az `SDL_BlitSurface()` egy téglalapot az egyik felületről (surface) a másikra másol. Ha a forrástéglalap `NULL`, akkor az egész felület lemásolódik. Az `SDL_BlitSurface()` az a függvény, ami a színkulcsot alkalmazza, és kihasználja az RLE-kódolást.

Összegzés

Az SDL csökkenti a linuxos játékprogramok fejlesztésének az idejét. Elég kicsi ahhoz, hogy az elsajátítása ne vegyen el túl sok időt az életünkből, viszont elég sokat tud ahhoz, hogy akár kereskedelmi alkalmazásokat is fejlesszünk a segítségével. Remélem, hogy a cikk és a Bounce forráskódja elegendő lesz hozzá, hogy SDL-es játékok fejlesztésébe is bele merjünk vágni.

A kapcsolódó címek, valamint a listák megtalálhatóak a 49. CD Magazin/SDL könyvtárában.

Linux Journal 2003. június, 110. szám

Bob Pendleton (Bob@ Pendleton.com)
Az első programozási feladata az volt, hogy játékokat írjon át HP miniszámítógépekről az UNIVAC nagygépre. Azóta is a számítógépes játékok szerelmese. 1981 óta dolgozik a Unix különféle változataival és a Linuxszal. Bob független programfejlesztő és szakíró.

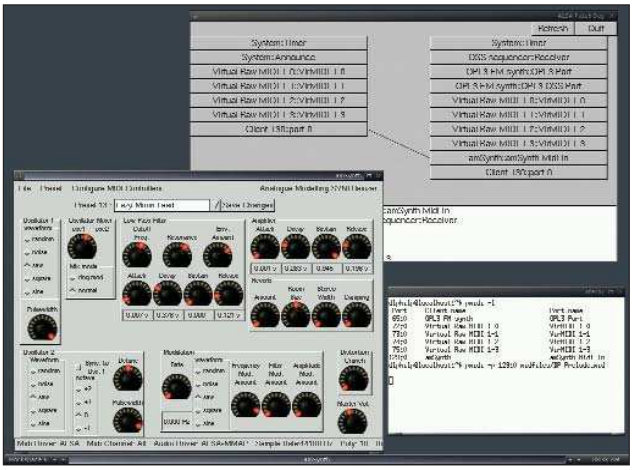
Szintetizátorok Linuxon

Varázsoljunk linuxos gépünkből szintetizátorstúdiót.

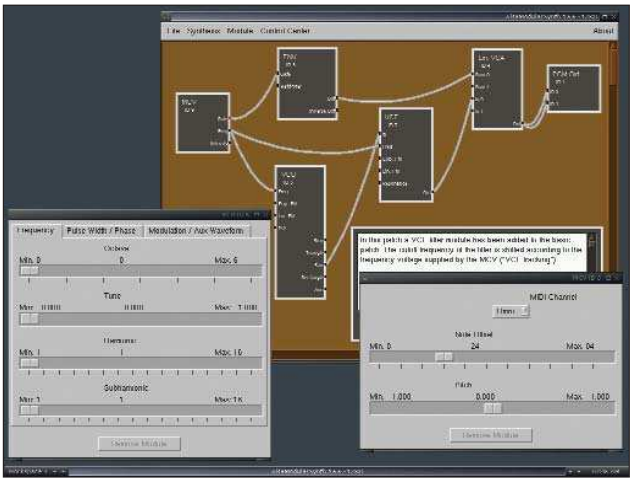
Az SWSS (software sound synthesis, vagyis programból megvalósított hangelőállítás) jelentős múlttal rendelkezik a számítógépek történetében. A digitális hangelőállítás korai kísérletei a Bell-laboratóriumban zajlottak, ahol a ***Max Mathews*** köré csoportosuló kutatócsoport 1969-ben megalkotta a Music V-ben csúcspontját elérő Music N hangszintetizáló programsorozatot. Azóta a Music V olyan neves digitális szintetizátorprogramok fejlesztési alapját képezte, mint a Csound, Cmix/RTCmix és a Common LISP Music. Ezek a programok jellemzően valamilyen nyelvkörnyezetet teremtettek felhasználóik számára, amelyekkel lehetővé vált a hangok tulajdonságainak, a hangjegyeknek és a hangmintáknak a hatékony leírása. A programok által használt nyelvekkel a felhasználók különválaszthatják a hangforrás (a hang előállításának elve) és a partitúra (például a kezdés pillanata, időtartam, összetétel jellemzői) megadását. A felhasználók az általuk használt SWSS-nyelven megkomponálják a hangszereiket és a partitúrát, majd az előállt kódot a nyelv fordítóprogramjának adják át. A kimenet egy fájlban keletkezik, ami bármilyen, az adott fájlformátumot támogató hangrendszeren lejátszható, vagy – megfelelő teljesítményű eszközök esetén – közvetlenül átirányítható egy digitális–analóg-átalakítóra, ami ennek alapján valós idejű hangleképezést tud előállítani a hangkimeneten.

A programban megvalósított szintetizátor (software sythesizer, softsynth) önmagában képes az imént vázolt modell partitúrarészének valós idejű vezérlésére. Ezek a szintetizátorok rendszerint vonzó grafikus külsővel rendelkeznek, gyakran utánozzák a valódi vezérlőpult megjelenését és kezelési jellemzőit, kiegészítőként pedig egy MIDI-billentyűzetre vagy egy külső sequencerre (sorrendvezérlőre) van még szükség az irányításhoz. Megfelelő körülmények között a szintetizátorprogram egy párhuzamos folyamat által is vezérelhető. Például az ALSA csatlakoztató segédprogramja segítségével a szintetizátorprogram hozzákötheto egy ugyanazon a gépen futtatott MIDI-sequencerhez. Így a hangsorozatok a szintetizátorprogramon keresztül rögzíthetők és játszhatók le, feleslegessé téve a külső szintetizátor használatát, ráadásul úgy, hogy még a MIDI-környezet is egyazon gépen a rendelkezésünkre áll.

A szintetizátorprogram alkalmazhat egy bizonyos hangszintetizáló eljárást (összeadó – additív, kivonó – szubtraktív, FM és így tovább), vagy nyílt végű és moduláris lehet. Az összeadó hangszintetizálási eljárás különböző hullámhosszú, amplitúdójú és fázisú szinuszgörbék összegeként állítja elő a kívánt hangot. A módszer rendkívül számításigényes és rettentő mennyiségű részadatot igényel az élethű hangzás eléréséhez. A kivonó módszer egy frekvenciákban gazdag hangmintát (például a fűrészfoghullám vagy valamilyen zaj) vesz alapul, majd bizonyos frekvenciák kiszűrésével alakítja ki belőle a kívánt hangot. A szubtraktív módszer viszonylag kis eszköz- és programigénnyel megvalósítható, s az így előállt hangok a 70-es évek analóg szintetizátorait idézik. Az FM szintézis az egyik oszcillátor által keltett rezgést alakítja át egy másik oszcillátor segítségével, így viszonylag kis számítási költségekkel is bonyolult hangsor előállítására nyílik lehetőség. Az FM-eljárást használó hangszerek közül a Yamaha DX7 szintetizátora a leghíresebb, és minden bizonnyal a cég OPL3 lapkája a legrosszabb hírű.



1. kép Az amSynth



2. kép Az ALSA moduláris szintetizátor (AMS)

A fizikai modellezés és a szemcsézett (granular) összetétel csak kettő napjaink újabb szintetizáló eljárásai közül. A fizikai modell módszere a valós vagy képzelt hangszer mechanikai tulajdonságait és működésének fizikai jellemzőit modellezi. Az eljárás jellemzői nem az olyan megszokott zenei mintákon alapulnak, mint a hullámformák, a frekvenciák és az amplitúdók, inkább fizikailag gerjesztett rendszerek jellemzőin, mint egy csőbeli légáram, a megpendített húr rezgése vagy a megütött membrán sugárirányú mintázata. A fizikai modellezés népszerű hangszintetizáló eljárássá vált, és többek között olyan cégek által kínált hangszerekben alkalmazzák, mint a Korg vagy a Yamaha. A szemcsézett módszer a hangadagok vagy -szemcsék

A linuxos szintetizátorok körképe

SOFTSYNTH	Változat	Grafikus felület	Beállítás	LADSPA	Jack	MIDI-jellemzők állítása	Felhasználói szerződés	Forráskód
amSynth	1.0-rc2	Gtk	Igen	Nem	Igen	Igen	GPL	Igen
Elara	1.1.1	X11	Igen	Nem	Igen	Igen	Tulajdonosi	Nem
Ultramaster Juno6	1.0.1	Gtk	Igen	Nem	Igen	Igen	Tulajdonosi	Igen
Bristol	0.9.1	X11	Igen	Nem	Nem	Nem	GPL	Igen
LegaSynth	0.4.1	Gtk	Igen	Nem	Igen	Igen	GPL	Igen
ALSA Modular	1.5.5	Qt	Igen	Igen	Igen	Igen	GPL	Igen
SpiralSynth Modular	0.1.0	FLTK	Igen	Igen	Igen	Igen	GPL	Igen
MSS	0.76.2	Gtk	Igen	Nem	Nem	Nem	GPL	Igen
AUBE	0.30.1	Gtk	Nem	Nem	Nem	Nem	GPL	Igen
RTSynth	1.7.0	FLTK	Igen	Nem	Igen	Igen	Tulajdonosi	Nem
JSyn	1.42	Java	n/a	Nem	Nem	Nem	Tulajdonosi	Nem
ZynAddSubFX	1.0.5	FLTK	Igen	Nem	Igen	Igen	GPL	Igen
gAlan	0.2.12	Gtk	Igen	Igen	Nem	Igen	GPL	Igen
jMax	4.0.0	Java	Igen	Nem	Igen	Igen	GPL	Igen
Pd	0.36	Tcl/Tk	n/a	Igen	Igen	Igen	BSD-szerű	Igen
Freebirth	0.3.2	Gtk	Nem	Nem	Nem	Nem	GPL	Igen
Ultramaster	2.0	Gtk	Igen	Nem	Nem	Igen	Tulajdonosi	Nem
iiwusynth	1.0.0	n/a	n/a	Nem	Nem	Igen	GPL	Igen
RX/Saturno	0.0.1	n/a	Nem	Nem	Nem	Nem	GPL	Igen

többé-kevésbé sűrű hanghalmokká való rendezésén alapul. Az eljárás jellemzői nem annyira szemléletesek, mint a korábbi eljárások esetén, de a módszer hatékony és a hangok igen széles skálája hozható létre vele. Még csak napjainkban próbál utat találni az olcsóbb sorozatgyártott szintetizátorokhoz, de hardveres megvalósítást már találhatunk a Kyma-rendszerben és az UPIC munkaállomásában.

Jellegzetes szintetizátorelrendezések

Egy szintetizátorprogram használhat csupán egyetlen szintetizáló módszert, lehet két vagy több eljárás keveréke, vagy választhatja a nyitottabb moduláris felépítést is. Mindegyik felépítésnek megvan a maga erőssége. Nagy vonalakban nézve talán a moduláris felépítés a legrugalmasabb, de a széles körű használhatóság oltárán a szabályozás (felbontás) minőségét esetleg fel kell áldozni. Az egymódszeres szintetizátorprogramok nélkülözik a moduláris felépítésűek rugalmasságát, de a jellemzőik rendszerint sokkal finomabb beállítását teszik lehetővé.

A moduláris szintetizátorok az elemekből való építkezés módszerét támogatják azáltal, hogy egyszerű különálló szintetizáló elemeket kínálnak, biztosítva tetszés szerinti egymáshoz kapcsolásukat. Például egy oszcillátor kimenetét egy burkológörbére irányíthatjuk vagy éppen fordítva. Ez a feketedoboz-szerű kapcsolatépítés jól alkalmazható a programból történő utánzásra, ahogy látni is fogjuk, amikor a cikk későbbi részében néhány moduláris szintetizátorral ismerkedünk meg.

Az általános célú programok közti különbségek egyre jobban elmosódnak. Példa erre a Csound, ami már FLTK alapú elemkészletet kínál a felhasználó által tervezett vezérlőpult elkészítéséhez. Számos felhasználó bonyolult felhasználói felületet hozott létre a Csound különböző eljárásainak kezeléséhez, ezek némelyike elég részletes ahhoz, hogy önálló Csound-alapú szintetizátorprogramként is megállja a helyét. Valószínűleg ez az irányvonal folytatódik a Common LISP Music és RTCmix SWSS-környezetekhez írt felhasználói felületek formájában is.

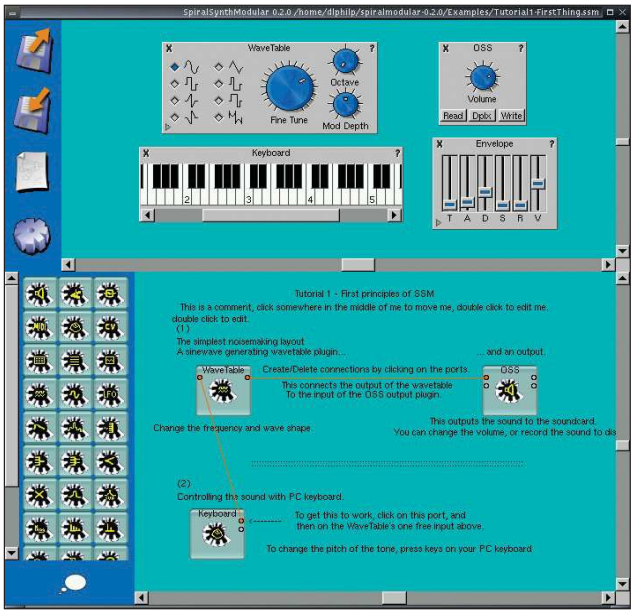
A jMax és a Pd által képviselt grafikus elrendezések további jelei ennek az összemosódási folyamatnak. Ezek szintén alkalmas elemkészletet kínálnak szintetizátor-kezelőfelületek előállítására, de – ellentétben a Csounddal – ezek az elemkészletek az eredeti munkakörnyezet részeként jelennek meg. A jMax és a Pd a grafikus és nyelvi elemek olyan egyedi kombinációját hasznosítja, aminek virtuális vezetékekkel való összekapcsolása révén jön létre a szintetizáló vagy feldolgozó hálózat. Ezek a környezetek is alkalmazhatók szintetizátorprogramként, de felhasználási céluk jellege miatt közelebb állnak a Csoundhoz, mint az itt ismertetendő szintetizátorprogramokhoz.

A beatbox stílusú szintetizátorok a szintetizátorprogramok egy újabb tervezési csoportját alkotják. Ezek a programok egy szintetizátor, egy dobgép és egy sequencer elemeit vegyítik egy „minden egyben” tartozékcsomaggá, bár a kifinomultabb megoldások lényegesen rugalmasabb zenei kompozíciós lehetőségeket kínálnak.

Ezek csak a főbb csoportok, de a cikk szempontjából ennyi elég ahhoz, hogy a szintetizátorprogramok alapvető típusait jelezzék. Az egyes hangszintetizáló módszerek és szintetizátorfelépítések részletes leírásait a ***Kapcsolódó címek*** (49. CD Magazin/Synth könyvtár) között felsorolt hivatkozásokon találhatja meg a téma iránt mélyebben érdeklődő olvasó.

Bővítmények

Ha már használtuk korábban az Adobe PhotoShop programját vagy a Gimpet, akkor már minden bizonnyal otthonosan mozgunk a bővítmények (plugin) világában. Az egyszerű felhasználó számára a bővítményeket használó programfelépítés a program képességeit anélkül terjeszti ki, hogy a programot frissítenünk kellene vagy újra kellene fordítanunk. A programozók számára lehetővé teszi, hogy figyelmüket a program lényeges tervezési kérdéseire összpontosítsák, és a bővített vagy továbbfejlesztett



3. kép A SpiralSynth Modular

4. kép Az SSM beállításai

5. kép Az RTSynth

6. kép A Bristol

tulajdonságokat a bővítmények hatáskörébe utalják.

A Windows, illetve Mac operációs rendszert használó zenészek a Steibert VST és Microsoft DirectX bővítményfelületeire írt bővítményeket használhatják. A Linux közvetlenül nem támogatja ezeket az API-kat (programozói felületeket), bár látni fogunk majd egy, a WINE segítségével működő közvetett eljárást. A linuxos zenefejlesztők saját, Linux-környezetre írt bővítményszerkezettel rukkoltak elő, aminek a neve Linux Audio Developers Simple Plugin Architecture (LADSPA, vagyis Linux zenei-fejlesztői bővítményszerkezet). A LADSPA API idővel szabvánnyá vált és támogatása ma már szinte minden új linuxos zenei programnak elvárt jellemzője. Létezik néhány kiemelkedő LADSPA bővítménygyűjtemény, amelyek nemcsak a szokásos hatásokat és DSP-ket foglalják magukban, hanem szintetizátor-építőelemeket (oszcillátorokat, burkológörbéket, szűrőket és így tovább), sőt néhány teljesen kialakított bővítményszintetizátort is. Ezenkívül létezik néhány jelentős nem LADSPA-bővítmény is.

Peter Hanappe iiwusynth nevű programja egy kisméretű szintetizátor, ami szintetizálómotorjához a SoundFontsot alkalmazza üzemanyagként. Az iiwusynth kimenete a SoundFonts megfelelő készletével nagyon jó, és számos program beágyazott szintetizátoraként népszerűvé is vált. Természetesen különálló szintetizátorként is használható a parancssorból.

A következő pehelysúlyú bővítményszintetizátor az RX/Saturno, ami a népszerű Yamaha DX7FM-szintetizátort utánozza. ***Juan Linietsky***, a program szerzője jelezte, hogy az RY/Saturno még mindig korai fejlesztési szakaszánál tart, de már így is elég hasznos és bővítményszintetizátorként alkalmazható minden olyan programban, ami támogatja az ALSA-sequencert.

Kjetil Matheussen vstserver nevű, figyelemre méltó programja a WINE képességeit arra használja, hogy megtévessze a VST-bővítményeket, amelyek így azt hiszik, hogy eredeti windowsos környezetben működnek. A legtöbb esetben a teljesítmény kiváló – legalább olyan jó, mint Windows alatt. Kjetil két ügyfélprogramot is írt a kiszolgálóhoz: egyet a VST-bővítmények Pd-hez való kapcsolásához, egyet pedig az LADSPA számára. A vstserver néhány VSTi-bővítményt is támogat, amelyek teljesen kidolgozott készülékek – szintetizátorok, mintavételezők és MIDI-sequencerek – VST-bővítmény szerkezetbe bújtatva.

Bár az LADSPA hatékony és népszerű szabvány, tervezési nézőpontból tapasztalható „egyszerűsége" lehetetlenné teszi a feldolgozás és a szabályozás néhány fajtáját. Maguk az LADSPA-bővítmények nem engedélyezik a MIDI-n keresztül történő közvetlen paraméter-ellenőrzést; jóllehet a bővítmények jól felhasználhatók az olyan MIDI-sequencerekben, mint a MusE. A Linux zenefejlesztői közössége megérett az új kihívásra, az XAP nevű új szabványtervezetre. Az API jelenleg még a tervezés szakaszában van, de az XAP-n dolgozó programozócsapatban az LADSPA tervezői és más tehetséges Linux-programozók is jelen vannak.

Az ALSA

A MIDI-bemenetet adó eszköz általában egy MIDI szintetizátorbillentyűzet, de bármilyen MIDI-hangszer használható. Ennek egy szabványos hangkártyához való csatlakoztatásához egy MIDI-csatlakozókábelre van szükségünk. Az OSS/Free és az ALSA támogatja az MPU-401-megfelelő eszközöket, így néhány különálló MIDI-kártya is működni fog. Az ALSA közvetlen támogatást nyújt a soros kapuhoz és az USB-n keresztül csatlakozó MDI-eszközökhöz (ezeket a kapcsolatokat nem próbáltam ki), s ezeken túlmenően a nagyon hasznos virmidi virtuális MIDI-kapukat.

A programoldalról tekintve az alap OSS/Free Linux-hangrendszer (a rendszermag hangrendszere) alkalmas az itt ismertetett

program szintetizátorral való együttműködésre, de a javasolt rendszer az ALSA programkönyvtárat és meghajtóprogramokat, a JACK audiocsatoló készletet és a hardveres MIDI bemeneti eszközt tartalmazza. A legjobb válaszidő a rendszermag kis késleltetési idővel – esetleg az időosztásos (preemptive) folttal – való fordításával érhető el. A valós idejű óra (real-time clock – RTC) engedélyezése is ajánlott.
A 2.5-ös rendszermagtól kezdve az OSS/Free hangrendszert hivatalosan az ALSA váltotta fel. A 2.6-os üzembiztos rendszermagtól kezdve az ALSA kerül a rendszermagba, ami kitűnő OSS/Free emulációval rendelkezik az ALSA-t nem ismerő alkalmazások számára. A 2.5-ös változatnál korábbi rendszermag az OSS/Free rendszert tartalmazza – akik ilyet használnak, saját maguk kénytelenek az ALSA-t fordítani és telepíteni. Ahogy az egy korszerű hangrendszertől elvárható, az ALSA is széles körű kapcsolódási lehetőségekkel bír, programozói felületet nyújt bővítmények írásához, fejlett hangügyfél-kiszolgáló felépítéssel, a rendszerbeállítások és felügyelet megkönnyítésére pedig hatékony eszközgyűjteménnyel rendelkezik.
A 4Front Technology cég OSS/Linux terméke szintén jól működik a linuxos szintetizátorokkal, bár nyilvánvalóan nem tudja kihasználni közvetlenül az ALSA-sequencer ügyfelek hálózatának előnyeit.

A JACK

A JACK a JACK Audio Connection Kit (JACK hangkapcsolati eszközkészlet) kifejezés önmagára hivatkozó rövidítése. Tervezésekor a cél egy rövid válaszidejű profi teljesítményű programból megvalósított csatolóeszköz létrehozása volt a munkafolyamaton kívüli hangprogramok számára. Rendeltetését tekintve hasonlít az olyan hangkiszolgálókhoz, mint a KDE-hez készült aRts, vagy a Gnome esd programja, de ezeknél nagyobb teljesítményű, a profi hangszabványokkal jobban együttműködő eszköz. A JACK adatcsatornát használó programok szabadon irányíthatják át hangbe- és kimeneteiket, akár összetett forgatókönyv szerint is, például egy MIDI által vezérelt programból megvalósított szintetizátor kimenetét egy merevlemezes felvevőre, mialatt modulált bővítményhanghatást is alkalmazunk, és mindezt valós időben, kis késleltetési idővel. Bár a JACK a Linux hangvilágának még viszonylag új szereplője, mégis számos fejlesztő és felhasználó figyelmét felkeltette, és nincs messze az a pillanat, amikor a bevezetése és a használata egyaránt magától értetődő lesz a Linux hangprogramozásával foglalkozók és az egyszerű felhasználók számára.

A tesztkörnyezet

A számítógép, amin a kipróbálást folytattam, egy 800 MHz-en futó AMD Duron processzorral, 512 MB memóriával és egy 15 GB-os IDE-merevlemezzel volt felszerelve. A gép hangrendszere két hangkártyából, egy SoundBlaster SBLive Value-ból és egy SoundBlaster PCI128-ból állt; MIDI-billentyűzetet, bemenetként pedig egy Casio CZ101 szintetizátort használtam. Szükségem volt továbbá ***Steve Ratcliff*** pmidi programjára a MIDI-fájlok lejátszásához; valamint egy második gépre, amin a Voyetra Sequencer Plus Gold programját futtattam MS-DOS alatt. A képmegjelenítéshez egy 19 colos monitort és Voodoo3-as videokártyát alkalmaztam. A hangkártyák kimeneteit egy Yamaha DMP7 keverőre kötöttem, innen egy 100 wattos QSC teljesítményerősítőre került a jel, amit egy Yorkville Sound YSM-10-es referencia-hangszórópár szólaltatott meg.
Az alacsony és középszintű programokat a 2.4.5-ös, alacsony válaszsebességre foltozott Linux-rendszermag, az ALSA 0.9.0rc6 programcsomag (audiokönyvtár, meghajtóprogramok és segédprogramok), a legfrissebb JACK, valamint ***Richard Furse*** és ***Steve Harris*** LADSPA bővítménye képviselte. A használt programok között volt még ***Maarten de Boer*** alsamixergui és ***Bob Ham*** ALSA MIDI pacth bay programja, amelyek többek közt grafikus felületet biztosítottak az ALSA alsamixer és aconnect segédprogramjaihoz.

Futtassuk-e a programból megvalósított szintetizátort rendszergazdai jogosultsággal?

Számos itt bemutatandó szintetizátorprogram leírása a program rendszergazdai jogosultsággal való futtatását javasolja, akár a rendszergazda nevében, akár a programnak rendszergazdai jogosultságot adva, a suid beállításával. Ez rendszerint nagyobb elsőbbséget (priority) biztosít a futó alkalmazásnak, ugyanakkor komoly biztonsági kockázatot jelent a hálózaton ügyködő felhasználó számára.
Eltekintve a biztonsági vonatkozásoktól, el kell mondanom, hogy amikor egy valós idejű folyamat kicsúszik a rendszergazda irányítása alól, az eredmény elég csúnya szokott lenni, s nem ritka, hogy a gép teljes lefagyásával jár. Az egyik próba során, amikor rendszergazdai jogosultsággal tevékenykedtem, egy egyszerű nem felismert MIDI-eszköz is lefagyasztotta a rendszert. Legyünk tehát óvatosak!

Körséta a programból megvalósított szintetizátorok körül

A Linux Sound & Music Software honlap Software Synthesis szakasza tartalmaz egy alfejezetet a programból megvalósított szintetizátorok és mintavételezők témájában (Softsynths & Samplers). Jelenleg több mint harminc működő hivatkozás vezeti be a látogatót a programból megvalósított szintetizátorok sokszínű világába. Az táblázat (lásd a 29. oldalon) ennek a sokféleségnek csupán egy szeletét mutatja be a szintetizátorok polifonikus képességeire (egyszerre több hang lejátszásának lehetőségére) összpontosítva, figyelmen kívül hagyva a hálózaton keresztül nem elérhető szintetizátorokat és környezeteket, mint amilyen a Csound és az RTCmix. A valós idejű természetüknek köszönhetően szerepeltetem a felsorolásban a beatbox-programokat, valamint a Pd és jMax MAX-szerű környezeteket.
Mivel írásomat összegzésnek szánom és nem az egyes programok szembeállításának, tovább ritkítom a táblázat sorait, s a következőkben a felsoroltak közül csak néhányat fogok ismertetni.

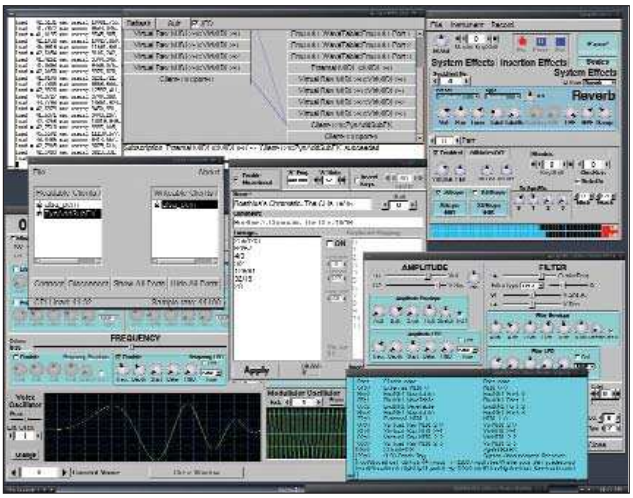
Az amSynth

Nick Dowell amSynth programja csupán egy a hangszintetizáló – mégpedig a szubtraktív – eljárást használó szintetizátorok közül, de szerintem a legkitűnőbb is. A jel a hagyományos utat követi, a két oszcillátor kimenete egy szűrőn és egy erősítőn halad keresztül, hogy az így kapott jel a digitális hanghatások (az amSynth torzításra és visszhangosításra is képes), illetve a modulációk lépésein keresztül haladva jusson a hangkártyánk digitális–analóg-átalakítójára (DAC). A hagyományos hangösszetétel három fő részegysége a feszültségvezérelt oszcillátor (voltage-controlled oscillator, VCO), a feszültségvezérelt szűrő (voltage-controlled filter, VCF) és a feszültségvezérelt erősítő (voltage-controlled amplifier, VCA).
Az amSynth néhány jól hangzó alapbeállítással rendelkezik. Nincs olyan összeállítás, ami támogatná a szabványosított MIDI hangszer-összerendelést, de a program értelmezi a bejövő MIDI Program Change üzeneteket. Mivel az amSynth elsősorban egy olyan önálló hangszer, ami csak egy MIDI-csatornát kezel egyszerre, a legjobban talán vezető hangszerként vagy csatlakozó felületként használható. MIDI-sequencerrel is meghajtható.
Az amSynth teljes neve Analogue Modelling Synthesizer.

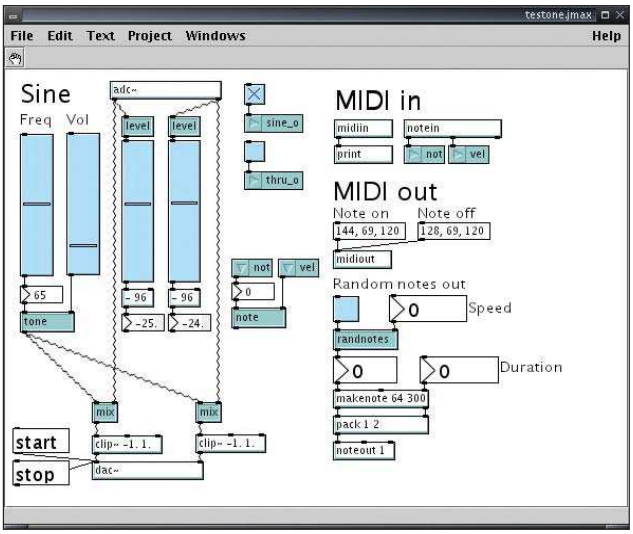
Nincsenek benne valódi feszültségvezérelt alkatrészek, így joggal tehetjük fel a kérdést, hogy vajon Nicknek sikerült-e elérnie célkitűzését, az analóg különbségi szintetizátor hangjának modellezését. Örömmel jelenthetem ki, hogy az amSyinth által előállított hangok teltek és élethűek. Nem kötelező elhinni a szavaimat, de az amSynth honlapján található kitűnő bemutatók (demo) sokkal jobban mutatják a hangját, mint amennyire én itt leírhatnám.

Az ALSA moduláris szintetizátor

Dr. Matthias Nagorni számos hasznos programot és segédprogramot írt az ALSA-ra, JACK-re és LADSPA-ra, ezek közül pillanatnyilag leginkább a nagyszerű ALSA Modular Synthesizer (AMS) tűnik ki. A program a közelmúlt nagyszerű moduláris szintetizátorainak működését utánozza, a felhasználók rendelkezésére álló elemek széles választékát biztosítva.
A *2. kép* az AMS legegyszerűbb formáját mutatja. Egy különbségi hangszintézis összeállításakor az adat útvonala alapjában véve megegyezik az amSynth által használttal, a különbség az AMS nagyobb rugalmasságában keresendő. Az amSynth rögzített kivitelezésével ellentétben az AMS teljes rugalmasságot mutat különféle moduljai összekapcsolási lehetőségeinek tekintetében. A legtöbb modul bármilyen bemeneti kapcsolatot készséggel elfogad, és csak kissé vagy egyáltalán nem törődik azzal, hogy a kimenetét mire irányítjuk. Azért nem árt vigyázni, ha a modulokat nem a megszokott módon kapcsoljuk egymáshoz, mert ilyenkor a kimenetek is elég váratlanok vagy túlvezéreltek lehetnek; az ilyen összeállítások kipróbálásánál mindig óvatosan kezeljük a hangerőszabályzót. Minden alkotóelem saját párbeszédablakkal bír (ami a 2. képen is látható), ez az elem nevén történő jobb egérkattintással csalogatható elő.
Dr. Nagorni az alábbi tanulságos megjegyzéseket mellékeli az AMS-hez: „az AMS az általa megvalósított különleges szolgáltatásaival biztosítja, hogy mindhárom fő hangszintetizáló eljárás (összeadó, különbségi, FM) könnyen használható legyen. A `Dynamic Waves` (változtatható hullámok) modul az additív hangszintézist valósítja meg akár nyolc oszcillátor egyetlen modulban történő használatával. Minden egyes harmonikus egy nyolcpontos burkológörbével formázható, a burkológörbék grafikusan megjeleníthetők. Az FM-eljárásnál hasznos egészfelharmonikusok finombeállítására a VCO-k (feszültségvezérelt oszcillátorok) egy további harmonikus- és alharmonikus-csúszkával bírnak. A szükséges lineáris FM bemeneti kapu is rendelkezésre áll. A különbségi hangszintézis megfelelő működéséhez elengedhetetlenül fontos, hogy a feszültségvezérlés a hagyományos 1V/oktáv szerinti logaritmikus szabály szerint működjön. Ilyen módon tetszőleges helyre kerülhet a szűrő levágási frekvenciája, a tökéletes VCF-követés mindenképpen megmarad. A logaritmikus frekvencia más szempontból is hasznos, például egy LFO-val (alacsonyfrekvenciás oszcillátor) történő vibrato (hangmagasság-ingadozás) esetén.
Az AMS-t valós idejű működésre tervezték. MIDI-vezérlésre különösen alkalmas, a legtöbb jellemzője MIDI-vezérlőegységhez csatlakoztatható és valós időben változtatható. Egyszólamú és többszólamú szintetizátorként egyaránt jól használható, és egyszerre több példányt futtatva a JACK-en keresztül multitimbral beállítás is megvalósítható. A LADSPA-bővítmények támogatásával már igen gazdag szolgáltatáskészlettel rendelkezik, ami az AMS-t eszményi választássá teszi azok számára, akiknek nincs szintetizátoruk. Egy teljes MIDI zeneszerző környezet építhető fel, amihez nincs szükség másra, mint egy elfogadhatóan gyors gépre, egy olyan remek Linux MIDI-sequencerre, mint a MusE vagy a Rosegarden, és az AMS-re. Egyes beállítások jobban, mások kevésbé jól működnek, így



7. kép A ZynAddSubFX



8. kép A jMax

a jó doktor kiadós készletet készített elő példa-összeállításokból tanulmányozás és kísérletezés céljából a felhasználó számára. Ezek közül néhány megtalálható az AMS honlapján elérhető bemutatófájlok között, de ahogy az itt bemutatott összes szintetizátorra elmondható, javaslom, hogy töltsük le és telepítsük fel magunknak, hogy az igazi képességeit láthassuk.

A SpiralSynth Modular

Először vala a SpiralSynth, ezt követte a SpiralLoops program, ez a dögös hurok-sequencer, majd lőn ezután a SpiralSynth-Baby mint a SpiralLoops bővítménye. Végül ***Dave Griffiths*** fejlesztő úgy döntött, hogy mindezt egyetlen nyílt végű moduláris szintetizátor-összeállító eszközkészletté rakja össze, amit SpiralSynth Modularnak (SSM) nevezett el. Az AMS-hez hasonlóan a SpiralSynth Modular is egy üres munkaterületből és a modulok választékából áll, amiket a felhasználó a munkaterületre helyezhet és összekapcsolhat, de az SSM saját egyedi felépítéssel és hangképző képességekkel rendelkezik.
A *3. kép* az SSM-et mutatja futás közben az első oktató-összeállításával. Ez a példa a hangszintézis egyszerű fajtáját mutatja, amit hullámtábla-szintézisnek is neveznek. A hullámtábla egy előre megadott és tárolt hullámforma (szinusz, négyszög, há-

romszög, impulzus és így tovább), amit a virtuális billentyűzet vezérel, és a burkológörbe-létrehozó módosít, még mielőtt az OSS kimeneti modulján keresztül a hangkártya DA-átalakítójára kerül. A példában láthatjuk, hogy a szintetizátor a számítógép billentyűzetéről lett megszólaltatva, de az SSM egy MIDI-modult is biztosít a MIDI-üzenetek fogadására és továbbítására.
A billentyűzetmodul is érdekes élményt nyújt, remek szórakozás volt noteszgépem Qwerty billentyűin keresztül az SSM-en játszani.
Az SSM nem működik natív ALSA-sequencerügyfélként, így nem lehet olyan módon közvetlenül egy ALSA-kapura kötni, mint az amSynthet vagy az AMS-t. Összeköthető viszont a szabványos OSS/Free MIDI-eszközével (***/dev/midi***), s így bármilyen erre az eszközre csatlakoztatott eszköz vagy program bemenete lehet. Ha a gépünkben nincs MIDI-készülék, használhatjuk az ALSA virmidi virtuális MIDI-kapujait, beállítva az SSM beállításai között a MIDI-csatornának a megfelelő kaput (***/dev/snd/midiC1D0*** a noteszgépem esetén, lásd a *4. képet*). Ez lehetővé teszi a más ALSA-t ismerő folyamathoz való csatlakozást az aconnecten vagy az ALSA patch bayen keresztül.
Dave Griffiths az SSM honlapján előzékenyen rendelkezésünkre bocsátja a szintetizátor néhány kitűnő bemutatóját is. A program FLTK-felülete kellemes és könnyen használható. Az SSM bőven tartalmaz érdekes és hasznos modulokat (ide értve az LADSPA-támogatást is), a legfrissebb változat pedig a JACK támogatásával is lefordítható. Dave a program közeljövőben megjelenő változatának SpiralLoops bővítményét nagymértékben továbbfejlesztett állapotban tervezi bemutatni, és várható, hogy az ALSA is több közvetlen támogatásban részesül majd.

RTSynth

Az egyik kedvenc szintetizátorom ***Stefan Nitschke*** RTSynth nevű programja – ez az összeállítható szintetizátorok egy újabb kitűnő példája. Adott a munkafelület, az ikonként megjelenő modulok a munkaterületre helyezhetők és összekapcsolhatók, a modulon jobb egérkattintásra előugró ablakban pedig az adott modul jellemzői állíthatók be. Az itt bemutatott programból megvalósított szintetizátorok közül az RTSynth az egyetlen, ami a hangokat a fizikai modell alapján hozza létre.
A fizikai modellezésen alapuló hangszintézis rendkívül valószerű hangok előállítására képes. Erről az RTSynth néhány összeállítása gyorsan meggyőzhet bennünket. Az RTSynt honlapja néhány bámulatos akusztikus és elektromos gitárhangot mutat be teljes hangszerelésben, basszussal, dobbal és billentyűsökkel. Az RTSynth egy multi-timbral képességgel rendelkező programból megvalósított szintetizátor dobeszközökkel kiegészítve. A bemutatók jól tükrözik e teljes megoldást nyújtó programból megvalósított szintetizátornak a képességeit.
Az RTSynth az ALSA-t és JACK-et egyaránt kezelni tudja. Teljes körű MIDI-támogatást nyújt az ALSA és a korábbi OSS/Free rendszermag-hangmodullal is. Ha a rendszerünk nem rendelkezik az ALSA-meghajtókkal, ebben az esetben is van lehetőségünk arra, hogy az RTSynthtel külső munkafolyamatokhoz, például egy párhuzamosan futó MIDI-sequencerhez csatlakozzunk, mégpedig a Unix nevesített csővezetékének a segítségével. A csővezeték egyszerű eljárást biztosít a folyamatok közötti adatcseréhez olyan programok esetén, amelyek esetleg nem rendelkeznek az adatmegosztás egyéb lehetőségével.
Az alábbi példa az RTSynth segítségével mutatja be a nevesített csővezeték használatát.
Először egy csővezetéket hozunk létre az `mkfifo` segédprogrammal:

```
mkfifo $HOME/tmp/midififo
```

Ezután az RTSynthet készítjük fel a csővezetékből érkező adatok fogadására:

```
RTSynth < $HOME/tmp/midififo
```

Végül pedig a nevesített csővezetéket elsődleges kimeneti eszközként kell megadnunk a működtető program számára. Példánkban ***Simon Kagedal*** virtuális billentyűzetét használtam erre:

```
clavier -o $HOME/tmp/midififo
```

Most már közvetlenül használhatjuk az RTSynthet a billentyűzetről. A következő paranccsal egy közönséges nem nevesített csővezetéket is használhatunk egy folyamat kimenetének az RTSynthre való irányításához:

```
cat foo | RTSynth
```

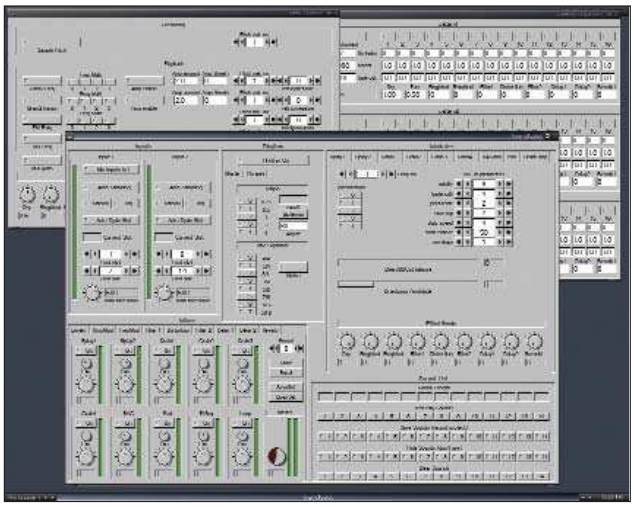
Ezek a csatlakoztatási módszerek különösen MIDI-eszköz, illetve az ALSA virmidi meghajtóprogram hiányában hatékonyak.

A Bristol

Nick Copeland neve leginkább talán az Slab merevlemezes rögzítőrendszeréről híres, de neki köszönhetjük a Bristol Synthesizer Emulator programot is. Ez a programból megvalósított szintetizátor grafikus felületet és hangelőállító egységet ad a Mini Moog, Moog Voyager, Sequential Circuits Prophet-5, Roland Juno-6 és Yamaha DX7 szintetizátorok utánzásához.
A palettáján ezeken felül szerepel még a Hammond B3 és Vox Contunental orgona, valamint a Fender Rhodes elektronikus zongora. A Bristol egy általános keverőpult és a Yamaha Pro10 digitális keverőjének utánzására is képes, de ismertetőm az ezzel kapcsolatos tapasztalataimat nem tartalmazza.
Mint a *6. képen* is látható, a grafikus felületek rajza igényes, de a program többet nyújt puszta látványosságnál. Nick, amennyire csak lehetséges volt, utánozta az eredeti szintetizátorokon található kezelőszervek működését. Jelenleg nincs még mindegyik szintetizátor összes tulajdonsága kidolgozva és Nick bevallása szerint némely emuláció (különösen a DX7 esetén) még egy kis csiszolást igényel, de az összes kapcsoló, forgatógomb és tárcsa valós időben egyenletes választ és gyors paraméterfrissítést adva kapcsolgatható, forgatható vagy tekergethető. A Bristol nem csupán a különféle szintetizátorok és billentyűzetek valósághű megjelenítésének félelmetes feladatát oldja meg, hanem emellett hangképző egységeiknek is élethű hangzású utánzatát adja.
A `./startBristol -v -h` paranccsal indítva a program listát ad a futásidejű beállításokról, amikkel a működés nagymértékben testreszabható. Például a

```
./startBristol -alsa -seq -bufsize 2048 -voices 6
```

paranccsal indítva a Bristol az alapértelmezett Mini Moog-módban indul, meghajtóforrásként az ALSA-t választva, emellett a parancs regisztrálja a Bristolt az ALSA-sequencerben, a hangkártya számára beállítja az átmeneti tár méretét (az alapértelmezett 1024, de Nick 2048-at javasol a SBLive kártyám számára), és hat csatornára korlátozza a szólamok számát (a Bristol alapértelmezésként 16 csatornát kezel). Mellesleg a Bristol több példányban is futhat párhuzamosan, lehetővé téve a szintetizátorok lépcsőzését – ahogy azt a régi időkben is tettük.
Sokkal több helyet igényelne, ha a Bristol minden felületét megfelelő módon be szeretném mutatni. A példa, amit a



9. kép Varga István csoundfltk programja Oeyvind Brandtsegg ImproSculptjának futtatása közben

➲ http://www.linux-sound.org/sounds címen tettem elérhetővé, csak a Mini Moog utánzását mutatja be, de ez alapján képet alkothatunk arról, hogy mit várhatunk ettől a szintetizátoregységtől – kellemes időtöltést az elmúlt idők szintetizátorainak világában!

Ultramaster Juno6

Ez a szintetizátor kitűnő példája az igazi szintetizátor utánzásának. A Juno6 billentyűi és a vezérlőpult kezelőszervei mind valósághű képet kaptak, és a Bristolhoz hasonlóan aktívak, és minden pillanatban készek a beavatkozásra. Korábban volt egy igazi Juno6-om, ennek alapján elmondhatom, hogy az Ultramaster emulációjának hanghűsége meglepően jó, mindez együtt a digitális moduláció megbízhatóságával. Mindent összevetve az arpeggiator (arpeggiatornak hívjuk azt az eszközt, ami egy MIDI-ről bejövő akkordot önálló hangokra bont és megadott sorrendben ritmikusan ismétli őket – a fordító) működik a legjobban. Azok számára, akik még emlékeznek az ilyen rokonszenves dolgokra, valószínűleg jó szórakozást fog nyújtani ez a szolgáltatás; sajnos az arpeggiatorok nem olyan megszokottak manapság, úgyhogy akinek ez újdonságot jelent, várhatóan érdekes és szórakoztató órákat tölt majd a használatával.
A Juno6 a különbségi hangszintézis tiszta megvalósítása, ami hatásos szűrők megszólaltatására is alkalmas. Az Ultramaster honlapján egy rövid wav formátumú példafájl is található, de többet tudhatunk meg a szintetizátor hangjáról valamint képességeiről, ha egyszerűen csak játszadozunk vele.

A ZynAddSubFX

Paul Nasca ZynAddSubFX programja az összegző és különbségi hangszintetizáló eljárás érdekes keveréke, a további feldolgozásra egy hanghatás-programrésszel kiegészítve. Ha csak ennyit nyújtana, már akkor is figyelemre érdemes lenne, de egy kitűnő FLTK-felület is csábít a hangszintézis különböző paramétereivel való kísérletezésre, és kedvenc MIDI-sequencerünkről ALSA-ügyfélként vezérelhető. A *7. kép* mutatja a ZynAddSubFX és a pmidi MIDI fájllejátszó együttműködését. Látható továbbá a ZynAddSubFX ***Scales*** (hangsorok) párbeszédablaka, amint a Scala programból érkező beállításokat fogadja. Egy új hangsor kiválasztásakor az éppen használt összeállítás beállításai azonnal frissülnek, ami szokatlan hangzások kipróbálására ösztönöz, érdekes változásokat hozva ismert zeneanyagokba.

A ZynAddSubFX is rendelkezik multi-timbral képességgel, MIDI-csatornánként szólhatnak meg a különböző hangszerek, s ezzel jó választási lehetőséget kínál a minden célra alkalmas, programból megvalósított szintetizátorok között (eltekintve a dobtól, sajnos). Hangja az egyszerű szintetizáló eljárásokat követi, de ezeknek a módszereknek a hatékony alkalmazása és a program kitűnő felülete együttesen szép hangok megszólaltatásában nyújt segítséget. Az összeállított művek közvetlenül a ZynAddSubFX-ben rögzíthetők, a fejlesztő pedig számos bemutatóművet elérhetővé tett, amivel megmutatja, hogy mire is képes hangszere önálló, multi-timbral programból megvalósított szintetizátorként. A bemutatott programból megvalósított szintetizátorok közül a ZynAddSubFX a legújabb, de a fejlesztése folyamatos. A cikk írása közben értesültem róla, hogy a program már képes a JACK kezelésére (7. kép), így a hangsorok támogatásával, a Scalából történő hangolás lehetőségével, az ALSA-sequencer ügyfélként való beállításával és a JACK csatlakozási lehetőségekkel ez a szintetizátor a korszerű linuxos hangprogramok jellegzetes képviselőjévé lépett elő.

A jMax és a Csound

Nem feledkeztem meg arról az ígéretemről, ami szerint az inkább kódalapú hangszintetizáló környezetek ismertetését elkerülöm, de megemlítettem azt is, hogy ezek fejlődése során egy összemosódási törekvés is megfigyelhető. A jMax gyors ütemben fejlődik sokoldalú zeneszerző, illetve -feldolgozó programcsomaggá, de SWSS-eszközkészletként is alkalmazható. A *8. kép* egy egyszerű jMax-összeállítást mutat, a saját maga által előállított leírással kiegészítve. A példa meglehetősen hétköznapi, a program ennél sokkal összetettebb összeállítások kezelésére is képes.
Varga István csoundfltk (Linuxra készített Csound csomag) programját a *9. kép* mutatja az ImproSculpt futtatása közben, ami egy valós idejű mintavételező meglehetősen bonyolult FLTK-felülettel. Ez a példa nem kimondottan szintetizátor-összeállítás, de jól példázza a Csound FLTK-eszközkészlet lehetőségeit, amelyekkel a felhasználók grafikus kezelőrendszereket és vezérlőpultot tervezhetnek a Csound hangszintetizáló és -feldolgozó modelljeik számára. Más példák a Csound egyszerű szintetizátorként való alkalmazását mutatják be; az érdeklődőknek a további példákért és anyagokért mindenképpen érdemes ellátogatniuk a
➲ http://www.csound.com oldalra.

Zárszó

A legjobb tanács, amit búcsúzásképpen adhatok, az az, hogy az itt bemutatott programból megvalósított szintetizátorok közül minél többet próbáljatok ki otthon. Írhatnék rajongó szólamokat, de az igazi próba csak az lehet, ha a saját fületekkel halljátok, hogy mire képesek ezek a programok. Úgyhogy előre, töltsétek le és telepítsétek, amelyik a bemutatott programok közül rokonszenves, és csapjatok egy kis élvezetes lármát. Én is fülelni fogok!

Linux Jorunal 2003. május, 109. szám

Dave Phillips
Az ohioi Findlayben élő zenész, tanár, író. 1995 óta, vagyis a Linuxszal való első találkozása óta lelkes tagja a Linux hangjával foglalkozó közösségnek. A Kiskapu Kft. kiadásában tavaly megjelent Zene és Hang című könyv szerzője.

Linuxszal a tudományos múzeumokért

Használjuk tudásunkat arra a nemes célra, hogy segítsük az újító, szórakoztatva oktató kiállításokat az új közönség megszerzésében.

Az elmúlt évek során fiammal szabadidőnk jelentős részét arra áldoztuk, hogy a New York állambéli Itchacában lévő Sciencenter számára a kiállítást bemutató programot készítettünk. Ez a cikk azt mutatja be, hogy mennyiben hasonlít ez a projekt egyéb programfejlesztési projektekhez, miben mutat eltérést, és mik voltak a Linux használatából adódó előnyök. Nemcsak a Linuxról, illetve a programról lesz szó, hanem azokról a fejlesztési folyamatokról is, amelyekkel egy ehhez hasonló környezetben való munka során találkozhatunk.

A célközönség

A tudományos múzeumot bemutató program tervezésekor a jellegzetes programfelhasználóktól némileg különböző közönségre kellett felkészülnünk, mivel a program más módon kerül felhasználásra. Korábbi, Microsoft Windows alapú kiállítóprogramjaim – a Measurement Factory és a Fabulous Features – a körülbelül hatéves gyerekekkel záródó korosztályt célozta meg, míg a Linux alapú kiállítások – a Sound Studio és a Traffic Jam – közönsége az ötven-hatvanévesek közül került ki. A programnak elég egyszerűnek kellett lennie ahhoz, hogy a célközönség megértse a képernyőn látottakat, és hogy a látvány miképpen kapcsolódik a kiállításhoz. Egy kiállításon a számítógép nem maga a kiállítás, útmutatóként vagy egyszerűen manipulálható eszközként szerepel a többi tárgy között. Számítanunk kell egy másodlagos „közönségre" is, ez pedig a múzeum személyzete. A teremben dolgozókat (akiknek jelentős része önkéntes) nem lehet minden egyes számítógép hóbortjaira felkészíteni. A kiállítás gépeinek és programjainak önműködően kell elindulniuk, amikor reggel működésbe hozzák a múzeumi rendszert, mint ahogy az intézménynek legalább néhány árammegszakító lekapcsolásáról is gondoskodnia kell az esti záráskor. Nyilván az ember azt szeretné, hogy a személyzet szeresse a kiállítást, hiszen ha megnehezítjük az életüket, akár az „átalakítás miatt zárva" táblát is kiakaszthatjuk az ajtóra.

1. kép A Traffic Jam a Montshire Museum of Science-ben (Montshire Tudományos Múzeum – Vermont, Norwich)

A verseny

Álmaink kiállításának tervezgetésekor a legfontosabb dolog, aminek állandóan a szemünk előtt kell lebegnie, az az, hogy amikor a fiatalabb látogatókat megcélozva, jól látható módon elhelyezünk egy számítógépet a múzeumban, versenyre kell kelnünk az összes játékkal, amit valaha számítógépen játszottak, és az összes televíziós vagy mozifilmmel, amit addig láttak. Kiállításunknak ezért messzemenőkig látványosnak, figyelemfelkeltőnek kell lennie. A Measurement Factory teljesíti ezt az elvárást, azáltal, hogy a látogatók megmérhetik a testsúlyukat, a magasságukat, a szorításuk erősségét, összehasonlíthatják az eredményeiket kortársaik eredményeivel, és befejezésként egy bizonyítványt is kapnak minderről.

A Traffic Jam (*1–3. kép*) szintén friss és szórakoztató, hiszen a látogatók játszhatnak a közlekedési lámpákkal, és elkerülhetik a forgalmi dugókat – vagy előidézhetik, ha épp ehhez van kedvük.

A Sound Studio (*4–5. kép*) a látogatók számára lehetővé teszi, hogy saját és barátaik hangját többsávos hangrögzítőre rögzítsék, majd visszajátszáskor olyan egyszerű hatásokkal lássák el,

2. kép Erős forgalom a Traffic Jamben

3. kép A közlekedési lámpák beállítása a Traffic Jamben

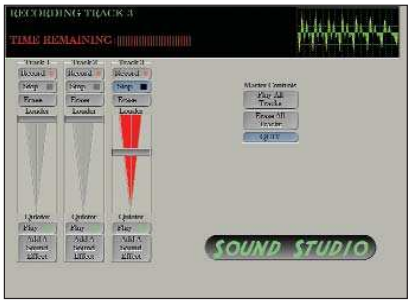
4. kép A Sound Studio

mint amilyen a visszhang.

Mindennek nyilván egyértelmű határai vannak. Például nem túl jó ötlet a „Hibáztál!" jelzést (hangot, képet, animációt) túl látványosra tervezni, különben a látogatók ezt tekintik majd céljuknak – soha ne akadályozzuk a kiállítás céljainak megvalósulását!

A fejlesztési folyamat

A célközönség és a nevelési célok szempontjából fontos a pontos fejlesztési folyamat kidolgozása. A következőkben



5. kép Hangfelvétel a Sound Studióval

ismertetett szempontok némelyike egészen hétköznapinak tűnhet, mégis fontos a betartásuk. Saját fejlesztési ciklusunknak természetesen nem kell pontosan megegyeznie az itt ismertetendő folyamattal.

- ***Az indító megbeszélés***: ha egy olyan különálló kiállítást szándékozunk létrehozni, ami nem része egy nagyobb, szervezett bemutatónak, akkor ez a szakasz akár el is maradhat. A Tech City például körülbelül egy tucat olyan egymással összefüggő kiállításból állt, amelyek a technika és a mérnöki tudományok központi témája köré szerveződtek. Ezért egy egész napos indító megbeszélést tartottak a projekt céljainak megvitatására és egy közös kezdeti nézőpont kialakítására. Több ilyen megbeszélésen részt vettem már, mindegyik bőven megérte a rá fordított időt. Amellett, hogy ráhangolódunk a projektre, hallhatunk egy-két jó ötletet, és találkozhatunk olyan emberekkel, akik segítségünkre lehetnek a folyamat későbbi szakaszaiban.
- ***Ötletbörze (brainstorming)***: egyrészről tedd a lábad az asztalra, és ne törődj még a Linuxszal sem, amíg eszedbe nem jut néhány igazán jó ötlet. Másrészről, ha tudomásunk van egy már létező alkalmazásról, amit esetleg egy kicsit módosítva hasznossá tehetnénk, említsük meg, és készítsünk jegyzeteket. Mialatt körbejárjuk a felmerülő ötleteket, tartsunk kéznél egy listát a projekt elvárásairól. Ezek lehetnek egészen egyszerűek, mint „egy műszaki tudomány bemutatása”, de kicsit bonyolultabbak is, például „szemléltessük a Heisenberg-féle bizonytalansági elvet általános iskolások számára egyszerű grafikus módszerrel, virtuális beavatkozási lehetőségekkel”. Mindig tartsuk szem előtt az elvárásokat, bár garantálhatom, hogy ezek az utolsó pillanatig változni fognak.
- ***Prototípus készítése***: ez a szakasz megegyezik más területek megfelelő szakaszaival, azzal a különbséggel, hogy esetünkben több szórakozást rejt magában, azoknak az embereknek köszönhetően, akikkel együtt dolgozunk. A múzeumok munkatársai, akikkel volt szerencsém együttműködni az évek során, kivétel nélkül értelmes, alkotóerőben gazdag, a munkájuk iránt lelkesedő emberek voltak. Ne döbbentsen meg minket túlzottan a kiállításhoz esetleg szükséges pavilon prototípusának a látványa. Valószínűleg igen sok kábelcsatornára és kartonlemezre lesz majd szükség.
- ***Az első ellenőrzés***: ez az alkalom, amikor a prototípusunkat először tesszük ki bírálatnak. Ne aggódjunk amiatt, ha a folyamatban érintett felnőttek nem bírálják olyan hatásosnak az alkotásunkat, mint amilyennek mi tartjuk. Ne feledjük, hogy a felnőttek egészen másképp gondolkodnak, mint a gyerekek. Minden kötöttség nélkül mutassuk meg a prototípusunkat néhány, a célközönség korában lévő gyereknek.
- ***Célteszt***: talán ez a kiállítás megalkotásának a legszórakoztatóbb része. Ez az első (és esetleg a második, harmadik) alkalom, hogy az alkotást a látogatók számára kipróbálásra üzembe helyezzük, így megfigyelhetők a válaszreakciók. Várhatóan többször is végrehajtjuk ezt a folyamatszakaszt. Néhány fontos kérdés, ami megfontolandó, illetve válaszra érdemes: vonzó a kiállítás a látogatók számára? Milyen hosszú időt töltenek a helyszínen a látogatók? Előfordul, hogy több látogató együtt használja a gépet? Megtanulják a látogatók, amit megpróbálunk nekik megtanítani? A megfelelő korosztályt szólítja meg a program?

1. lista Az állandóan futó program indításának pszeudokódja

```
A CD-ROM befűzésének kísérlete a techcity
saját könyvtárában a merevlemez sajátkönyvtára felett.
Nincs szükség a visszatérési kód ellenőrzésére.
A "reboot monitor" folyamat elindítása.
Belépés "techcity" néven és a startx futtatása
Ha a CD befűzése sikeres:
  A .xinitrc leállítja a reboot monitort, letiltja a
  képernyőkímélőt és az energiatakarékos üzemmódot,
  elindítja a programot.
különben:
  A .xinitrc lefuttat egy Python-parancsfájlt, ami
  a felhasználót a program-CD behelyezésére kéri,
  majd vár, amíg a felhasználó a "done" gombra kattint.
  Ez utasítja a reboot monitort a rendszer újraindítására.
A CD-ROM leválasztása.
```

2. lista Az S99xx-mytechcity fájl, ami a /etc/rc2.d fájlba lett beillesztve a program önműködő újraindulása érdekében

```
mount -o ro,user,exec /dev/cdrom /home/techcity
/root/rebootmon &
su - techcity -c "cd /home/techcity;startx"
umount /dev/cdrom
```

3. lista A Sound Studio .xinitrc fájlja

```
echo -n "Q" >/tmp/rebootfifo

xset s noblank
xset s off
xset -dpms

cd SoundStudio-newmeter/src
exec ./soundst
```

4. lista A járműleíró rész a Traffic Jam beállításfájljának dokumentumtípus-meghatározó állományából

```
<!ELEMENT vehicles ((vehicle)+,colors)>
  <!ATTLIST vehicles numtypes NMTOKEN #REQUIRED>
  <!ELEMENT vehicle EMPTY>
    <!ATTLIST vehicle length       NMTOKEN       #REQUIRED
                      maxspeed     NMTOKEN       #REQUIRED
                      accel        NMTOKEN       #REQUIRED
                      slowdistance NMTOKEN       #REQUIRED
                      stopdistance NMTOKEN       #REQUIRED
    >
  <!ELEMENT colors (color)+>
    <!ATTLIST colors num NMTOKEN #REQUIRED>
    <!ELEMENT color EMPTY>
      <!ATTLIST color red          NMTOKEN       #REQUIRED
                      green        NMTOKEN       #REQUIRED
                      blue         NMTOKEN       #REQUIRED
      >
```

Esetleg lehetőségünk nyílik elbeszélgetni azokkal a látogatókkal, akik kipróbálták a bemutatónkat, vagy ezt a múzeumi személyzet is megteheti, és tájékoztathat a benyomásairól. Talán ez a legértékesebb útmutató, amit kapni fogunk; a gyerekek szerencsére kegyetlenül őszinték.

- *Szakmai kiértékelés*: léteznek olyan cégek (USA), amelyek megfelelő szakértelemmel rendelkeznek oktató vagy múzeumi programok kiértékeléséhez. Ha a kiállításunk a Nemzeti Tudományos Alapítvány (National Science Foundation, NSF) valamely tagjának támogatásával létesül, akkor valószínűleg közülük is értékelni fogja valaki. Nekünk ezek közül csak eggyel, az Inverness Research Associatesszel volt alkalmunk kapcsolatban állni, de az észrevételeiket különösen hasznosnak találtuk. Ha lehetőségünk van rá, fogadjuk is meg ezeket a tanácsokat, sokat tanulhatunk belőlük.
- *Átdolgozás, beleértve a programkód letisztázását*: a prototípuskészítéshez hasonlóan ez a lépés is olyan, mint más fejlesztési folyamatok esetén az ennek megfelelő szakasz. Érdemes ilyenkor időt szánnunk a megjegyzések kibővítésére, a zavaros kód kigyomlálására.
A múzeumi programnak egyre több mindennel kell foglalkoznia. Ha valamikor a jövőben a program módosítása szükséges lesz, az a többletidő, amit most ezekre a dolgokra fordítunk, bőven megtérül. Ez a megfelelő alkalom arra is, hogy a kiállítás nem számítógépes részének is megépítsük a végleges változatát.
- *Az üzembe helyezés*: ez a szakasz megint csak hasonló az egyéb projektek megfelelő szakaszához, bár jó észben tartani, hogy az ilyen típusú projektek-nél fontos elvárás, hogy a rendszer folyamatosan működésre kész legyen, és ne igényeljen karbantartást. Ha a programunk egy nagyobb rendezvény részét fogja alkotni, ne feledjük, hogy a személyzet ilyenkor rettentő elfoglalt, ezért tegyük a lehető legegyszerűbbé a teendőiket. A Tech City rendszerének üzembe helyezése végül is nem állt másból, mint a gépek üzembe helyezéséből – négy ponton plusz egy tartalék –, a Linuxnak minden gépre történő feltelepítéséből és egy korábban előkészített CD-ROM-mal való tesztelésből (erről a Tippek és trükkök részben lesz még szó).

A gépek kiválasztása

Előfordulhat, hogy a programunknak egy már nem használt, régi gépen kell futnia, de az is megeshet, hogy a legkorszerűbb gépen, ami éppen a projekt számára nyújtott adomány része, vagy ha szerencsések vagyunk – legalábbis a Linux meghajtóprogramjainak a szemszögéből nézve –, kaphatunk egy olyan gépet, ami egy nemzedékkel idősebb. Ha abban a helyzetben vagyunk, hogy meghatározhatjuk a szükséges gép kiépítését, az elsődleges szempont a megbízhatóság legyen. A megfelelően kipróbált és kiforrott alkatrészek valószínűleg fontosabbak lesznek, mint a legújabb, leggyorsabb műszaki csodák, és forró pillanatoktól menthet meg egy valamivel lassabb processzor és videokártya, amikor hat hónap elteltével a távoli múzeumban egy meleg nyári napon nem következik be az üzemzavar. A rendszer hűtésével nem szabad takarékoskodni. Ha tehetjük, jó minőségű ventilátorokat válasszunk, és bárki is építi az esetleges kiállítási pavilont, biztosítson megfelelő légáramlatot a számítógépnek, és gondoskodjon a levegő elvezetéséről is. A Tech City programja adományként érkezett Hewlett-Packard Vectra 400-as gépeken lett üzembe helyezve, amelyek 1 GHz-es Pentium III-as processzorral, i810-es lapkakészlettel, és 20 GB merevlemezzel voltak felszerelve. Mindegyik gépet egy-egy Sound Blaster 16 PCI hangkártyával bővítettük ki, mivel a Sound Studio futtatása igényelte a teljesen kétirányú (full duplex) hangkártyát, vagy kettő olyat, ami nem rendelkezik ezzel a lehetőséggel. Mindegyik rendszer 19"-os monitorokkal volt felszerelve, szintén a Hewlett-Packard jóvoltából.

Az eszközök és a programkönyvtárak kiválasztása

Mivel nem ismerem az adott projektet, nem javasolhatok adott eszközöket és programkönyvtárakat. Arra sem teszek kísérletet, hogy valamilyen programozási nyelvet ajánljak, ellenben adhatok néhány olyan útmutatást, ami a munkánk során jó szolgálatot tett.
Tapasztaltabb fejlesztőknek biztosan nem lesz új, amit mondok, de ha egy már létező programcsomag közel áll ahhoz, amit az adott feladat esetén meg kellene valósítanunk, gondoljuk meg, hogy nem érdemesebb-e inkább azt módosítani és alkalmazni. A Traffic Jam felhasználói felülete négy részre osztotta az ablakot (a forgalom megjelenítése, a sűrűség beállítása, irányítás, egyéb adatok). Választásunk a kiterjedt témalehetőségek okán a GTK+-ra és az icewm ablakkezelőre esett, bár az utóbbit később kis mértékben módosítanunk kellett.
Természetesen a kölcsönözni kívánt kódra vonatkozó felhasználói szerződést figyelmesen el kell olvasnunk. Tartsuk tiszteletben a szerző jogait, és ne hárítsunk a múzeumra semmilyen kötelezettséget. Ahogy a gép kiválasztásánál, itt is az a legjobb, ha kellően körültekintőek vagyunk. Ha biztosak vagyunk benne, hogy nem lesz szükségünk a segédprogram vagy könyvtár legfrissebb változatának szolgáltatásaira, ne siessünk a telepítésével – a már ismert hibákat mindig könnyebb kezelni, mint a rejtetteket.
Ne feledkezzünk meg a karbantartásról és a projekt élettartamáról, hiszen vagy nekünk, vagy valaki másnak esetleg a jövőben módosítania kell majd a programot. A programozási nyelv vagy könyvtár egy még nem kibocsátott változata üzembiztosnak és jól működőnek tűnhet az adott pillanatban, de két év múlva esetleg teljesen más lesz vagy valamilyen

5. lista A járműleíró rész a Traffic Jam beállításfájljából

```
<vehicles numtypes="3">
  <vehicle length="10" maxspeed="10" accel="4"
           slowdistance="20" stopdistance="8"/>
  <vehicle length="15" maxspeed="10" accel="2"
           slowdistance="30" stopdistance="8"/>
<vehicle length="20" maxspeed="10" accel="1"
           slowdistance="40" stopdistance="8"/>
  <colors num="9">
    <color red="255" green="204" blue="51"/>
    <color red="204" green="153" blue="255"/>
    <color red="255" green="204" blue="204"/>
    <color red="255" green="153" blue="204"/>
    <color red="153" green="255" blue="204"/>
    <color red="204" green="204" blue="153"/>
    <color red="255" green="153" blue="51"/>
    <color red="153" green="204" blue="204"/>
    <color red="0" green="255" blue="255"/>
  </colors>
</vehicles>
```

szempontból nem működik együtt. Még a g++ is változott az alatt a négy év alatt, ami a Traffic Jam prototípusának elkészítése és a kész program üzembe helyezése között eltelt.
Végül kényelmetlen helyzetbe hozva magam még azt is bátorkodom javasolni, hogy a Linux-rendszercsomag választásakor is tartsuk magunkat bevált óvatosságunkhoz. A legújabb és legcsillogóbb változat alkalmas lehet asztali gépünkre, de a megbízhatóság ebben az esetben sokkalta fontosabb. Végső rendszerünkhöz a mi választásunk a Debian 3.0r0 változatra esett, ugyan nem sokkal a kibocsátása után, de mivel a Debian híres a visszafogottságáról, nem nyugtalankodtunk a döntés miatt.

Tippek és trükkök

Szeretnék megosztani néhány, a munkánk során felmerült nehézséget és az erre alkalmazott megoldási módszert. Lehet, hogy nem találtuk meg a lehetséges legjobb megoldásokat, de a mi esetünkben jól beváltak.
Feladat: *Hogyan tarthatnánk a rendszerünket mindig működésre kész állapotban? A kiállítási programnak a gép bekapcsolása után, felhasználói beavatkozás nélkül el kell indulnia. Szükség esetén a múzeumi személyzetnek képesnek kell lennie a program frissítésére.*
Megoldás: A mi esetünkben ez az volt, hogy a CD-ROM-on lévő alkalmazáskönyvtárat a kiállítás saját könyvtárának (például */**home**/**techcity***) legfelső szintjére fűztük be, és indításkor önműködően a megfelelő felhasználói néven léptünk be. Ha nincs megfelelő CD-nk – minden lemez csak egy kiállítás programját tartalmazza –, a képernyőn megjelenik a felszólító üzenet, hogy tegyük be a megfelelő lemezt, és indítsuk újra a gépet. (Bár a látogatók számára nem hozzáférhető, de a billentyűzet minden számítógéppel ott van a pavilonban.) Az újraindítást figyelő program egy FIFO kimenetét figyeli, hogy az R billentyű lenyomása esetén újraindítsa a gépet, vagy Q esetén lépjen ki, bár más megoldáson is gondolkoztunk.
Az *1. lista* a folyamat peszeudokódos leírása, a *2. listán* az önműködő újraindítást lehetővé tevő fájl látható, a *3. lista* pedig példát mutat a ***.xinitrc*** fájlra.
Feladat: *Az ilyen típusú programok általában igénylik a rendszer finomhangolását. Hogyan valósítsuk ezt meg?*
Megoldás: Első gondolatunk egy, a Windows ***.ini*** fájljához hasonló formátumú beállításfájl alkalmazása volt. Ez működött volna a Sound Studio esetében, de a Trafic Jamnél már nem. Az utóbbi többek között igényelte volna a különböző járműtípusok meghatározását, ami az XML képességeivel könnyen ábrázolhatóvá tette volna az azonos osztályba tartozó különböző egyedeket.
A fiam egy olyan C-könyvtárat írt, ami az xmllib2 felett futva lehetővé teszi, hogy az egyes alkotóelemekre faszerkezetként – vagyis az alkotóelem dokumentumbeli elérési útvonalával – hivatkozzunk.
A *4. lista* a Traffic Jam beállításfájljának dokumentumtípus-meghatározó állományából (DTD) mutat egy szakaszt, az *5. listán* pedig a fájl hivatkozott része látható. A 6. lista (49. CD Magazin/Museum könyvtár) a jármű fizikai tulajdonságait betöltő kódrészletet mutatja – látható, ahogy a Cfg a korábban említett könyvtáreljárásokkal körbevett C++-objektumra hivatkozik.
Feladat: *hogyan kezeljük az ablakkezelőt biztonsági szempontból? A látogatóknak nem szabad, hogy ki tudjanak lépni a programból, vagy más programokat tudjanak elindítani, hogy mozgatni tudják az ablakokat és így tovább.*
Megoldás: A korai tesztek során azt tapasztaltuk, hogyha a látogatók számára megengedjük a billentyűzethez való hozzáférést, előbb-utóbb megtalálták a módját, hogy kilépjenek a programból. Ezt a beállításfájlok és az icewm módosításának segítségével úgy oldottuk meg, hogy ne legyenek felbukkanó menük, és tiltva legyen az ablakok mozgatása, illetve átméretezése. Másrészt egyik program sem igényli a billentyűzet használatát, normál üzemelés közben az a pavilon belsejében maradhat.

Összegzés

Lehet, hogy nem az oktatóprogramok írása jelenti számunkra vágyaink netovábbját, de ha mégis így lenne, próbáljunk egy olyan műszaki múzeumot találni, amelynek segíthetünk. Szeretik az új arcokat, mindig szívesen fogadják a segítséget, és azt is garantálhatom, hogy többet kapunk vissza, mint amennyit mi adunk.
Szeretnék köszönetet mondani a New York állambeli ithacai Sciencenter munkatársainak és önkénteseinek egy óriási múzeum működtetéséért, és hogy az embereket gondolkodásra serkentik.
Látogassunk el a honlapjukra a ➲ http://www.sciencenter.org címen.

A fotók tulajdonosa a Sciencenter (Itacha, New York).

A cikkhez kapcsolódó listák megtalálhatóak a 49. CD Magazin/Museum könyvtárában.

Linux Journal 2003. június, 110. szám

Len Kaplan
(lkaplan@nlzero.com)
Akkoriban kezdett programozni, amikor a kisszámítógépek még hűtőszekrény nagyságúak voltak. A Linux mellett nagyon érdeklik a beágyazott rendszerek, szabad idejében szívesen foglalkozik vasútmodellezéssel.

Nyomtatottáramkör-tervezés Linuxszal

Linuxot használva már sikerült megírnunk egy mikrovezérlő programot. Ezek után most az áramköri kártyát is magunk tervezzük meg.

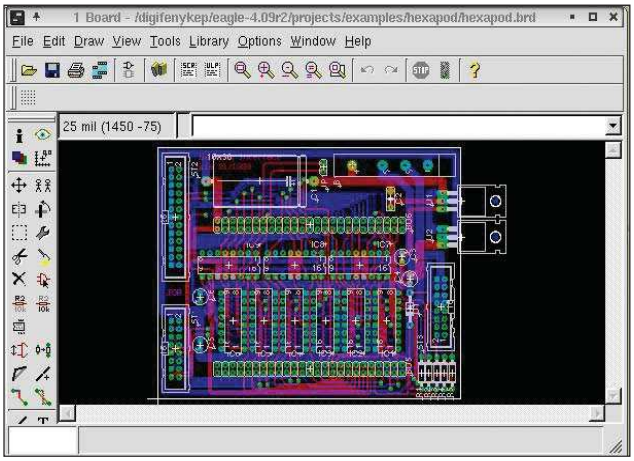
A Linux a legnagyobb hátrányban a tervezőprogramok terén van – ez az általános vélekedés. Így joggal tarthatunk attól, hogy a nyomtatott áramkör tervezéséhez szükséges CAD programok esetén is nehézségeink lesznek. Szerencsére ebben is megfigyelhető némi javuló tendencia, két irányból is: egyrészt bizonyos programozók már elkezdték írni az igényeiket kielégítő tervezőprogramokat (belefáradva abba, hogy nincsenek ilyenek); másrészt néhány cég alapvető céljául tűzte ki, hogy a programja a lehető legtöbb rendszeren, köztük Linuxon is fusson.
Először röviden egy, ebbe az utóbbi kategóriába tartozó programot említenék: az EAGLE nyomtatottáramkör-tervezőt. Talán nem véletlen, de ezt a programot egy német cég fejleszti. Németországban ugyanis ma már mind a kormányzat, mind más intézmények gépein is Linux fut (például a banki rendszereken). Hátha a hazai Linux (UHU) megjelenése is ebbe az irányba mozdítja a magyar felhasználást.
A program ismertetőjét a ➲ http://www.cadsoft.de oldalon találhatjuk (angol és német nyelven). A *freeware* hivatkozásra kattintva eljuthatunk az ingyenesen használható, ám ennek ellenére nem bemutatóváltozatot tartalmazó oldalához. Nem bemutatópéldány, mert mindent tud, amit a teljes változat, de az alábbi korlátozásokkal:

- A kártya mérete legfeljebb 100×80 mm lehet.
- Legfeljebb kétrétegű kártyákat lehet vele tervezni.
- A kapcsolásirajz-szerkesztő csak egyoldalas rajzokat kezel.
- Csak az online fórumon lehet hozzá (esetleg) támogatást kapni, telefonon és elektronikus levélben nem.
- Csak nonprofit felhasználást engedélyeznek regisztráció nélkül (kedvtelésből és oktatási célra megfelelő).

Letöltés után egy Linuxon futtatható programot találunk, amit végrehajtva megtörténik a telepítés. Nem rendszergazdaként is lehetséges telepítenünk az EAGLE-t, ekkor a felhasználó saját könyvtárába települ.
Amiért különösen érdemes kipróbálni ezt a programot (azon kívül, hogy létezik teljes értékű Linux-változata), az az, hogy a teljes értékű változatok ára sem elérhetetlenül magas (mint néhány CAD program esetében azt megszokhattuk). Ha például az ingyenes verzióval megegyező Light változatot szeretnénk regisztrálni, az 49 dollárba, a Standard változat 199 dollárba, a profi pedig 399 dollárba fog kerülni.
Milyen előnyökkel jár egy ilyen „gyári” CAD program használata a később ismertetendő, ingyenes, nyílt forrású programmal szemben?

- Beépített kapcsolásirajz-szerkesztőt tartalmaz, amiből közvetlenül előállítja az összekapcsolt pontok listáját (netlist).
- Az `autorouter` lehetőség is megtalálható benne, azaz el tudja készíteni (esetleg több-kevesebb hiánnyal) a nyomtatott áramkör tervét.
- Kimeneti formátuma olyan, hogy a legtöbb nyomtatott-áramkör-gyártó fel tudja használni a gyártáshoz, így ha nem magunk akarjuk elkészíteni a kártyát, akkor is minden készen áll a megrendeléshez.
- Regisztráció esetén támogatást kaphatunk a fejlesztőktől (esetleg a könyvtárak esetében is kiegészítéshez juthatunk: manapság naponta jelennek meg új áramkörök. Ez utóbbit – mivel még nem regisztráltam – nem próbáltam ki.).

Hátránya nyilvánvalóan az, hogy gyorsan ki lehet nőni a „kedvtelés” kategóriát, és akkor fizetni kell érte, ha nem is sokat.
Mint régi szabadprogram-rajongót, engem is – akárcsak valószínűleg az olvasók többségét – az érdekelt, hogy fellelhető-e nyílt forrású megoldás az adott feladatra. Nos, én a legérettebbnek, a gyakorlatban leginkább használhatónak a ***Thomas Nau*** által fejlesztett, nemes egyszerűséggel csak PCB-nek



nevezett programot találtam. Ő ugyan néhány évvel ezelőtt abbahagyta a fejlesztését, azóta ***Harry Eaton*** kezelte a programot – bár az ő honlapján se lehet túl sűrűn változásokat tapasztalni. Ezzel együtt a program jól használható, és mivel a forráskód nyílt, bárki nekifoghat a továbbfejlesztésnek.
Ha esetleg nincs rajta Linux-terjesztésünk CD-in (számoson szerepel), akkor a ➲ http://bach.ece.jhu.edu/~haceaton/pcb oldalról elindulva letölthetjük. Jelenleg az 1.6.3-as változat tűnt az utolsónak. Telepítése a szokásos módon zajlik (`./configure ; make ; make install`).

Tekintsük át, melyek a ***főbb előnyei***:

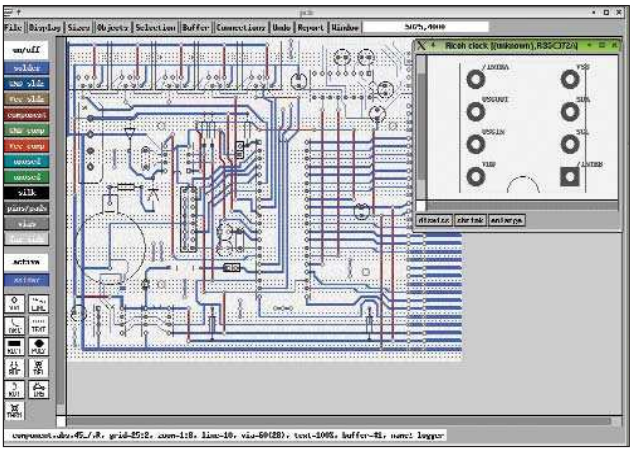
- Nyílt forrású, ingyenes.
- Nyolc rétegig működik.
- Jelentős méretű és bővíthető elemkönyvtárral bír.
- A Linux (Unix) alatt megszokott PostScript formátumú kimenete van, így könnyen nyomtatható, illetve átalakítható más formátumba (például pdf).
- Gerber-kimenet is van rajta, amit egyes kártyagyártók fel tudnak dolgozni.
- A leírása HTML formátumú.

Hátrányai:

- Gyakorlatilag nincs hozzá támogatás.
- A kártyagyártók nem biztos, hogy örülnek annak, ha csak Gerber- vagy PostScript-állományt tudunk adni.
- Nincs `autorouter`, azaz minden vonalat nekünk kell elhelyeznünk.
- Nincs beépített kapcsolásirajz-szerkesztő, tehát a netlist előállítása többletmunkát igényel; vagy ha nem is készítünk kapcsolási rajzot, később dokumentációs gondjaink lehetnek (elfelejtjük, hogy melyik kapcsolat miért van).

Azért, hogy ezzel a programmal megkönnyítsem a munka megkezdését, néhány (általam alkalmazott) módszert, műveletet ismertetnék. A program a `pcb` paranccsal indítható, ekkor egy üres tervvel indul el. Lássuk a kidolgozás néhány lépését!

- Először be szoktam állítani a kártya méretét. Ez azért célszerű, mert így mindig látjuk a „határainkat". Később, ha nagyon kell, lehet módosítani (általában növelni kell). Ezt a programablak felső részén látható menügombok közül a ***sizes***-ra kattintva tehetjük meg. Ennek ***PCB width***, illetve ***PCB height*** mezőjében állítható be a méret mil-ben, azaz egyezred collban (ez elég gyakran alkalmazott mértékegység az elektronikai alkatrészeknél). Ugyanitt lehet beállítani az aktív vonal, valamint az átforrasztás méreteit is, illetve a szöveg nagyságát.
- Maradjunk még egy kicsit a ***sizes*** menüpontnál! Itt négyféle huzalozási stílus közül lehet választani, illetve a hozzájuk tartozó vonal-, forrasz-szem és furatméreteket beállítani. Kényelmes, ha ezeket (signal, power, fat, skinny) az általunk kedvelt értékekre előre beállítjuk, és akkor később, munka közben csak ki kell választani a megfelelőt. Egyébként az előző pontban említett helyen megváltoztathatjuk az aktív méreteket, anélkül, hogy ezeket a beállításokat módosítanunk kellene. Figyelem, a kártyára már felhelyezett elemek nem módosulnak attól, hogy itt változtatunk! A tervben mindig az éppen beállított méretek jelennek meg, és nem a stílusok.
- A ***display*** menüpontban lehet megadni a kívánt rácsosztást, és hogy mutassa-e a rácsot (mint a pontokat) a terven. Ha lehetséges, a legkisebbalkatrész-lábosztásnál kisebbre szoktuk választani, hogy elég finoman lehessen vezetni a vonalakat. Felületforrasztott alkatrészek esetén esetleg megegyezhet a legkisebb lábosztással (ott úgysem fér el vonal a lábak között). Ugyanitt választható ki a nagyítás mértéke. Figyelem, miután kiválasztottuk, még kattintanunk kell a terven ahhoz, hogy alkalmazásra kerüljön! Azon a ponton kattintsunk, amelyiket szeretnénk, hogy a helyén maradjon. Ehhez képest lesz nagyobb, illetve kisebb a kártya többi része.
- A ***file*** pontban lehet menteni (vagy ha már létezik, betölteni) a terveket, itt tudunk nyomtatni is. Tudnivaló, hogy itt nem történik a megszokott, „valódi" nyomtatás, hanem csak fájlba írás. Ki lehet választani, hogy PostScript vagy Gerber formátumban írja-e ki a terveket, és hogy mi legyen rajta (körvonal, igazító jelek, fúrási segédjelek), valamint hogy milyen legyen a kimenet: színes, tükrözött, forgatott, invertált stb. Ezután egy másik programmal (PostScript esetén például a `gv`) megnézhetjük az eredményt. Ha magunk akarjuk elkészíteni a kártyát, akkor a `gv`-ből is kinyomtathatjuk a megfelelő rétege(ke)t. (Fotóeljárással készítendő áramkörnél célszerű a végleges változatot feketében pauszpapírra, lézernyomtatóval nyomtatni, ez használható maszknak a fotózásnál).



- A bal szélen lévő színes gombokkal választhatjuk ki, hogy melyik réteget akarjuk látni, az alatta lévő felbukkanó menüvel (az ***active*** feliratú címke alatti gombra kattintva jelenik meg) pedig azt, hogy melyikre akarunk elemeket elhelyezni. Leggyakrabban a ***solder*** és a ***component*** réteget alkalmazzuk.
- Bal oldalt lent helyezkednek el az eszközkiválasztó gombok (átforrasztás, vonal, ív, szöveg, négyszög stb.), általában a bal egérgombbal tehetjük le őket a kártyára. Vonal esetén, ha nem az előzőt akarjuk folytatni, hanem újat szeretnénk kezdeni, a CTRL gomb lenyomása közben az új vonal első pontján kattintsunk.
- A jobb gombbal kattintva jelölhetünk ki egy elemet. A jobb gombbal négyszöget húzva kijelölhetjük a benne foglalt elemeket. Ha üres helyen csináljuk, ezzel törölhetjük az addigi kijelöléseket.
- A középső gombbal (kétgombos esetben mindkettővel egyszerre) mozgathatjuk az elemeket.
- Ha egész kijelölést akarunk mozgatni, akkor a ***buffer*** menüből válasszuk ki a ***cut selection to buffer*** pontot, majd egy kattintással a referenciapontot, az előbbit ezután az utóbbinál fogva tetszőleges helyre mozgathatjuk, és a bal gombbal lerakhatjuk. Ha a ***copy selection to buffer*** pontot választjuk, ugyanígy egész részleteket is másolhatunk.
- Az adott eszközt vagy kijelölést, esetleg elemet az ESC billentyű lenyomásával dobhatjuk el.
- A beépített alkatrészkönyvtár a ***window*** menü ***library*** pontjával érhető el.
- Az ***objects*** menüben szerkeszthetjük a rétegeket, valamint itt lehet megváltoztatni az elemek nevét, és lekérni a lábkiosztást (ez egy külön ablakban jelenik meg).
- A ***selection*** menüből indulva módosíthatjuk a kiválasztott elemek méreteit (például utóbb jövünk rá, hogy mégis jobb lenne vastagabb vonal), és néhány egyéb tulajdonságát.
- A CTRL-M gombbal jelet tehetünk le. A jobb felső sarokban az ettől a jeltől való távolságot is mutatni fogja, nem csak a kártya bal felső sarkától mértet.

Nos, indulásnak talán ennyi elég is. Még egy dolgot emelnék ki, ami jóllehet benne van a leírásban, nekem némi fejtörést okozott – ez pedig a saját elem létrehozása. Nálam főleg csatlakozók esetében fordult elő, mivel annyi fajta csatlakozó létezik, hogy képtelenség mindet felvenni a könyvtárba, viszont célszerű elemként kezelni őket. Egyrészt, hogy lássuk a körvonalaikat, és hogy milyen közel lehet tenni hozzájuk más elemeket, másrészt pedig, hogyha több is van belőle, akkor elemként

nagyon gyorsan felpakolhatjuk őket a tervre.
Hogyan járjunk el? A következő lépések útján hozhatjuk létre az elemet:

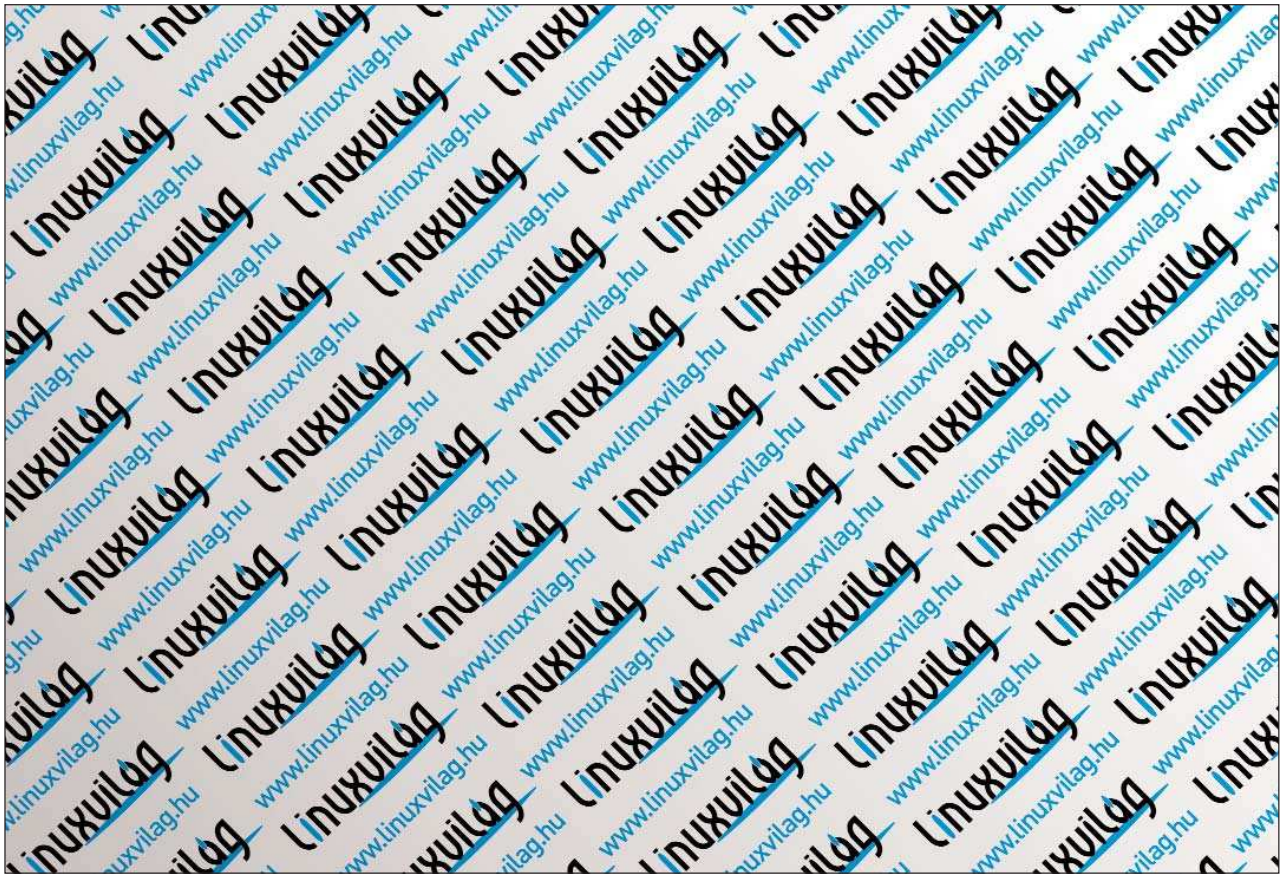
- Rakjuk fel a megfelelő átforrasztásokat (a lábak helyét).
- Egy vékony vonalat választva rajzoljuk meg a körvonalat.
- Jelöljük ki az így kialakult alkatrészt (a jobb gombbal húzva).
- Válasszuk a ***buffer*** menüből a ***cut selection to buffer*** pontot.
- Kattintsunk a kívánt referenciapontra (IC-knél egyes láb szokott lenni, másutt igény szerint).
- Válasszuk a ***buffer*** menüből a ***convert buffer to element*** pontot.
- Ezután a frissen létrejött elemet helyezzük el a kártyán.

Ha nagyon alaposak akarunk lenni (és hosszabb idő múlva is meg szeretnénk érteni a saját tervünket), akkor – az állományt mentve – a fájlba a lábak neveit is beleírhatjuk. Íme egy példa:

```
Element(0x00000000 "Ricoh clock" ""
➥"RS5C372A" 1380 1600 1 100 0x00000000)
(
   Pin(1550 1650 60 28 "/INTRB" "1" 0x00000101)
   Pin(1550 1550 60 28 "SCL" "2" 0x00000001)
   Pin(1550 1450 60 28 "SDA" "3" 0x00000001)
   Pin(1550 1350 60 28 "VSS" "4" 0x00000001)
   Pin(1250 1350 60 28 "/INTRA" "5" 0x00000001)
   Pin(1250 1450 60 28 "OSCOUT" "6" 0x00000001)
   Pin(1250 1550 60 28 "OSCIN" "7" 0x00000001)
   Pin(1250 1650 60 28 "VDD" "8" 0x00000001)
   ElementLine (1600 1700 1600 1300 10)
   ElementLine (1600 1300 1200 1300 10)
   ElementLine (1200 1300 1200 1700 10)
   ElementLine (1600 1700 1450 1700 10)
   ElementLine (1350 1700 1200 1700 10)
   ElementArc (1400 1700 50 50 180 180 10)
   Mark (1550 1650)
)
```

A fájlban az elemet (ha már több van) a helyzete alapján találhatjuk meg, ugyanis az elsősorban a szöveges jellemzők utáni két szám az elem helyzete mil-ben.
Miket írhatunk itt be? Az `element` paraméterlistájában a három idézőjeles szövegbe lehet beírni az alkatrész nevét, csoportját és típusszámát. Lejjebb a pin- (láb) leírásoknál a két szöveges részbe írhatjuk be a láb nevét és számát. A számát egyébként – ha ide nem írunk semmit – a bevitel sorrendjéből állapítja meg.
Ezek után visszatöltjük a tervet, és az ***objects*** menüből kiválasztjuk a ***display pinout*** pontot, majd az elemre kattintunk, és egy külön kis ablakban máris láthatjuk a lábkiosztást.
Sajnos annak a kérdésnek a tárgyalása, hogy miként készítsük el a megtervezett kártyát, meghaladja e cikk kereteit. Mindenesetre már így is sokkal közelebb kerültünk ahhoz, hogy működő mikrovezérlős rendszerünk legyen. Ha pedig az EAGLE programot választjuk, akkor akár profi termelőmunkát is végezhetünk kedvenc Linux-rendszerünkben.

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.



A linuxos USB alrendszer (2. rész)

A bemeneti alrendszer használata során nem számít, hogy egy bemeneti eszköznek hány gombja van, vagy hogy hányféle esemény létrehozására képes – most már felhasználói szintből kezelheted őket.

Sorozatunk előző részében láthattuk, hogyan működik a Linux bemeneti alrendszer a rendszermagban, végül még az eseménykezelőkre is kitértünk. Valójában minden kezelő másfajta felhasználói szintű programozói felületet nyújt. A különböző bemeneti eseményeket egy bizonyos formátumba alakítják át, ami az adott programozói felület használatával érhető el.

A bemeneti alrendszer rendszermagba építésének egyik kulcspontja magának az eseményrétegnek a megléte. Az eseménykezelő a feldolgozatlan eseményeket karakteres eszközcsomópontokon keresztül juttatja el a felhasználói szintre – minden egyes logikai eszközre egy saját karakteres eszközcsomópont jut. Az eseménykezelő felület nagyon sokoldalú módszer, mivel a segítségével a felhasználói szinten lehetővé válik az események kezelése, anélkül, hogy bármiféle információ elveszne. Például az ősi egértípusok csak két tengelyt támogattak, és legfeljebb öt gombjuk lehetett; ezek ténylegesen két tengelyre lettek leképezve, és csak három valódi gombra, mivel a 4. és 5. gombokat a görgetőhöz tartozó fel és le irányoknak feleltették meg.

Ez azonban gondot okoz akkor, ha egy olyan egeret próbálunk meg használni, aminek háromnál több gombja van, és emellett még egy görgetővel is bír, mivel a további gombok csak a már létező gomboknak feleltethetők meg. A korábbi programozói felület a fejlettebb bemeneti eszközök használatát sem tette lehetővé. Ilyen eszköz például az űrgolyó, vagy bármilyen hozzá hasonló több tengelyű eszköz is. Ezzel szemben az eseményalapú programozói felülettel tetszőleges eszköz összes lehetőségét ki lehet használni. Az eseményalapú programozói felület eszközönkénti listával rendelkezik az eszközök képességeiről és jellemzőiről.

Írásunk az eseményréteg különféle `ioctl` képességeit elemzi, ami az alapvető írási és olvasási hívásokat egészíti ki.

1. lista Példa egy EVIOCGVERSION függvényre

```
/* Az ioctl() meghívja a szükséges
   eszközmeghajtót */
if (ioctl(fd, EVIOCGVERSION, &version)) {
    perror("evdev ioctl");
}

/* az EVIOCGVERSION ioctl() egy egész számot
 * ad vissza amit kicsomagolunk és
   megjelenítünk */
printf("evdev driver version is %d.%d.%d\n",
       version >> 16, (version >> 8) & 0xff,
       version & 0xff);
```

2. lista Az input_id szerkezet meghatározása

```
struct input_id {
    __u16 bustype;
    __u16 vendor;
    __u16 product;
    __u16 version;
};
```

3. lista Egy EVIOCGID ioctl

```
/* néhány eszközadat lekérdezése */
if(ioctl(fd, EVIOCGID, &device_info)) {
    perror("evdev ioctl");
}

/* Az EVIOCGID ioctl() input_devinfo
 * szerkezettel tér vissza -
 * (a <linux/input.h> fájlnak megfelelően)
 * Így végigmegyünk az egyes elemeken,
 * és megjelenítjük mindegyiküket.
 */
printf("vendor %04hx product %04hx version
➥%04hx",
       device_info.vendor,
       ➥device_info.product,
       device_info.version);
switch ( device_info.bustype)
{
 case BUS_PCI :
     printf(" is on a PCI bus\n");
     break;
 case BUS_USB :
     printf(" is on a Universal Serial
            ➥Bus\n");
     break;
...
```

Az eseményréteg változatszámának kiderítése

Az eseményréteg támogatja az esemény eszközkód-változatszámának lekérdezését az `EVIOCGVERSION` `ioctl` függvény felhasználásával. A függvény értéke egy 32-bites egész szám (`int`) típus, aminek a felső két bájtja a változatszám első részét, a harmadik bájtja a változatszám második részét, az alsó bájtja pedig a kiegészítő változatszámot tartalmazza. Egy számítógépen minden eseményalapú eszköz ugyanazzal az értékkel tér vissza. Az `EVIOCGVERSION` alkalmazására az *1. listában* láthatunk

4. lista Csonkított karakterláncok

```
int fd = -1;
char name[256]= "Unknown";

if ((fd = open(argv[1], O_RDONLY)) < 0) {
    perror("evdev open");
    exit(1);
}

if(ioctl(fd, EVIOCGNAME(sizeof(name)), name)
➥< 0) {
    perror("evdev ioctl");
}

printf("The device on %s says its name is
➥%s\n",
       argv[1], name);

close(fd);
```

5. lista Az EVIOCGPHYS és a kialakítási adatok

```
if(ioctl(fd, EVIOCGPHYS(sizeof(phys)), phys)
➥< 0) {
    perror("event ioctl");
}
printf("The device on %s says its path is
➥%s\n",
       argv[1], phys);
```

6. lista Egyedi azonosító megállapítása

```
if(ioctl(fd, EVIOCGUNIQ(sizeof(uniq)), uniq)
➥< 0) {
    perror("event ioctl");
}

printf("The device on %s says its identity is
➥%s\n",argv[1], uniq);
```

példát. Az `ioctl` függvény első értéke egy megnyitott fájlleíró az eseményalapú eszközcsomópontra (például a */dev/input/event0*-ra). Az `ioctl` függvény harmadik értékeként nem magát az egészszámtípust kell átadni, hanem egy rá hivatkozó mutatót.

Az eszköz azonosítójának megállapítása

Az `EVIOCGID` `ioctl` segítségével az eseményréteg lehetővé teszi a kapcsolódó eszköztől adatok lekérdezését. A függvény értéke egy mutató az `input_id` szerkezetre, ennek felépítése a *2. listában* látható. Az `__u16` kizárólag a Linuxra jellemző, előjel nélküli, 16 bites adattípust jelöl. Saját programjaidban az `__u16`-ot `uint16_t`-re alakítva is nyugodtan használhatod. A busz típusa az egyetlen olyan mező, amelyik pontos adatot tartalmaz. Megfontolandó, hogy ez a típus egy kötött, sorszámozott típust használjon, amit a `<linux/input.h>`-ban

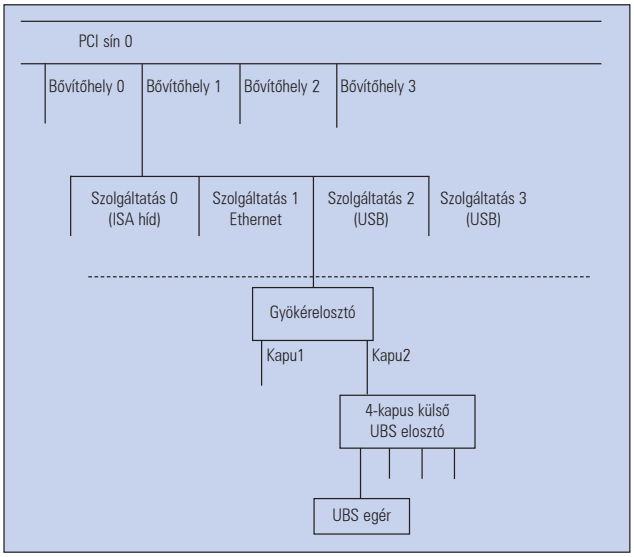
8. lista Események beolvasása

```
/* beolvasott bájtok száma */
size_t rb;
/* az események (egyszerre legfeljebb 64) */
struct input_event ev[64];

rb=read(fd,ev,sizeof(struct input_event)*64);

if (rb < (int) sizeof(struct input_event)) {
    perror("evtest: short read");
    exit (1);
}

for (yalv = 0;
     yalv < (int) (rb / sizeof(struct
     ➥input_event));
     yalv++)
{
    if (EV_KEY == ev[yalv].type)
        printf("%ld.%06ld ",
               ev[yalv].time.tv_sec,
               ev[yalv].time.tv_usec,
        printf("type %d code %d value %d\n",
               ev[yalv].type,
               ev[yalv].code,
               ev[yalv].value);
}
```



A billentyűzet topológiája

található `BUS_x` típus-meghatározókkal hasonlíthatunk össze. A ***gyártó***, az ***eszköz*** és a ***változatszám*** mezők az alkalmazott busz típusától függenek, az eszköz azonosítójának megfelelően. A jelenleg alkalmazott fejlett eszközök (főként az USB- és PCI-eszközök) ezeket az adatokat tartalmazzák, de az olyan örökölt eszközök, mint a soros egerek, a PS/2-es billentyűzetek vagy az ISA alapú hangkártyákon található játékkapuk (game ports) sajnos nem adnak erre vonatkozóan adatokat. Néhány busztípus esetében azonban ezek a számok értelmetlenek.

9. lista Egy írási függvény

```
struct input_event ev; /* the event */

/* első lépésben minden kijelzőt lekapcsolunk */
ev.type = EV_LED;
ev.code = LED_CAPSL;
ev.value = 0;
retval = write(fd, &ev, sizeof(struct input_event));
ev.code = LED_NUML;
retval = write(fd, &ev, sizeof(struct input_event));
ev.code = LED_SCROLLL;
retval = write(fd, &ev, sizeof(struct input_event));

while (1)
{
     ev.code = LED_CAPSL;
     ev.value = 1;
     write(fd, &ev, sizeof(struct input_event));
     usleep(200000);
     ev.value = 0;
     write(fd, &ev, sizeof(struct input_event));

     ev.code = LED_NUML;
     ev.value = 1;
     write(fd, &ev, sizeof(struct input_event));
     usleep(200000);
     ev.value = 0;
     write(fd, &ev, sizeof(struct input_event));
}
```

Az `EVIOCGID ioctl` használatára a *3. listában* láthatunk példát. Ez a függvény meghívja az `ioctl`-t, majd kiírja az eredményt. A `switch` szerkezet tartalmaz minden lehetséges busztípust; a program egy lehetséges kimenete: `vendor 045e product 001d version 0111 is on a Universal Serial Bus`. A busz típusán, a változatszámon és a gyártón (vendor), valamint az eszközt (product) jelölő számokon kívül néhány eszköz ezekhez tartozó karakterláncokat is ad, amelyek tartalmazhatják az eszköz és a gyártó pontos nevét. Ezeket a neveket az eseményrétegtől az `EVIOCGNAME ioctl` hívással lehet lekérdezni. Ez az `ioctl` egy karakterlánccal, és egy a karakterlánc hosszát jelölő egész számmal tér vissza (vagy egy negatív előjelű hibaazonosítóval). Ha a karakterlánc túl hosszú, a függvény levágja a karakterlánc végét. Az `ioctl` használatára a *4. listában* láthatunk példát. Talán feltűnt, hogy a függvénynek átadott kapcsoló nem `&név` formátumú. A magyarázat egyszerű: ha tömbre hivatkozunk, a tömb neve mindig egy mutató a tömb legelső elemére. Ha itt is `&név` alakban adnánk meg a tömböt, akkor valójában egy mutatót adnánk át, ami az első elemmutatóra mutat. Ha azonban mégis ragaszkodunk a `&név` formátumú megoldáshoz, akkor a `&(név[0])` alakot használhatjuk. Lássunk egy példát az eseménykód futtatására!
A */dev/input/event0*-ra befűzött eszköz elárulja a nevét: Logitech USB-PS/2 Optical Mouse.
Nem minden eszköz tartalmaz azonban használható neveket, ezért a rendszermagbeli bemeneti meghajtók valamilyen jelentéssel bírót próbálnak meg adni. Olyan USB-eszközök esetén, amelyek nem tartalmaznak a gyártóra vagy a készülék nevére utaló karakterláncokat, a bemeneti eszközmeghajtó összefűzi a gyártó és az eszköz azonosítóját, és azzal tér vissza.

Amellett, hogy az eszközazonosító és a névadat gyakran hasznos, olykor mégsem biztosítanak elegendő adatot arra nézvést, hogy milyen eszközzel is akadt dolgunk. Ha például két ugyanolyan botkormányod van, csak úgy tudod azonosítani őket, ha tudod, hogy melyik kapura csatlakoznak. Ezeket kialakítási adatoknak szokás nevezni, és az `EVIOCGPHYS ioctl` segítségével érhetők el.
Az `EVIOCGNAME`-hez hasonlóan ez is egy karakterlánccal tér vissza, valamint a hozzá tartozó hosszal (hiba esetén a negatív előjelű hibakóddal). Használatára az *5. listában* láthatunk példát.
A program lefuttatása a következőhöz hasonló kimenetet eredményez:

A /dev/input/event0-ra befűzött eszköz azt mondja, hogy az ő elérési útja usb-00:01.2-2.1/input0.

Hogy a fenti karakterláncot megérthessük, előtte darabokra kell szednünk. Az `usb` rész magától értetődően azt jelenti, hogy egy USB-eszközzel van dolgunk. A `00:01.2` érték az USB-vezérlő PCI-buszon lévő azonosítóját takarja, pontosabban a `00` számú a PCI-busz `01`-es rekeszének 2. feladata. A 2.1 a vezérlőtől az eszközhöz vezető utat mutatja, ahol a befűzött vezérlő az elsődleges vezérlő második rekeszébe kerül, az eszköz pedig a befűzött vezérlő második rekeszébe. Az `input0` azt jelenti, hogy az adott eszközön ez az első eseményeszköz. A legtöbb eszköz egyetlen ilyen eseményeszközzel bír, azonban például a multimédia-billentyűzetek a fő billentyűzethez tartozó eseményeket az első eseményeszközön továbbítják, a multimédia-művelethez tartozó eseményeket pedig egy második eseményeszközön adják tovább. Az ehhez tartozó példát az *ábrán* láthatjuk.
Ez az elrendezés nem nyújt megoldást, ha két azonos típusú eszközhöz tartozó kábelt cserélsz fel. Ebben az esetben csak akkor létezik megoldás, ha az eszközökhöz létezik valamilyen egyedi azonosítószám, például sorozatszám. Ezt az adatot az `EVIOCGUNIQ ioctl` hívással kérdezheted le, amire a *6. listában* láthatsz példát. A legtöbb eszköznek azonban nincs ilyen azonosítója, ilyenkor az `ioctl` üres karaktersorozattal tér vissza.

Az eszköz képességeinek és tulajdonságainak megállapítása

Néhány alkalmazás esetén elegendő az eszközazonosítót ismerni, mivel így – az eszköz fajtájától függően – minden eshetőség kezelésére képes lennél, a méretezhetőség szempontjából viszont ez nem megfelelő. Vegyük például, hogy engedélyezni szeretnéd a görgő használatát, de csakis abban az esetben, ha az egér rendelkezik görgővel. Nem lenne jó, ha a görgőt használó programunkban minden olyan gyártót és egértípust felsorolnánk, amelyik görgővel rendelkezik.
Az eseményréteg e nehézség elkerülését azzal teszi lehetővé, hogy megengedi annak megállapítását, hogy egy adott eszköz milyen képességekkel és jellemzőkkel bír. Az eseményréteg által támogatott tulajdonságok a következők:

- `EV_KEY` – meghatározott bináris értékek, például a billentyűk és az egérgombok.
- `EV_REL` – viszonylagos értékek, például az egér tengelyének elmozdulása.
- `EV_ABS` – meghatározott egészszámértékek, például a botkormány vagy a digitábla tengelyei.

10. lista Az eszköz gombjainak és billentyűinek mindenkori állapota

```
uint8_t key_b[KEY_MAX/8 + 1];

memset(key_b, 0, sizeof(key_b));
ioctl(fd, EVIOCGKEY(sizeof(key_b)), key_b);

for (yalv = 0; yalv < KEY_MAX; yalv++) {
   if (test_bit(yalv, key_b)) {
        /* a bit be van kapcsolva */
        printf("  Key 0x%02x ", yalv);
        switch ( yalv)
            {
            case KEY_RESERVED :
                printf(" (Reserved)\n");
                break;
            case KEY_ESC :
                printf(" (Escape)\n");
                break;
            /* other keys / buttons not
               shown */
            case BTN_STYLUS2 :
                printf(" (2nd Stylus
                ➥Button )\n");
                break;
            default:
                printf(" (Unknown key)\n");
            }
   }
}
```

- EV_MSC – egyéb dolgok, amelyek nem férnek bele egyetlen másik csoportba sem.
- EV_LED – a LED-ek és a hasonló kijelzők.
- EV_SND – hangkimenet, például a hangszórók.
- EV_REP – engedélyezi a billentyűk önműködő ismétlését a bemeneti magban.
- EV_FF – erő-visszacsatoló hatások küldése az eszköznek.
- EV_FF_STATUS – az eszköz által visszaadott jelentés az erő-visszacsatoló hatásokról.
- EV_PWR – energiagazdálkodási események.

Ezek a képességtípusok. Minden egyes típushoz további altípusok egész csoportja tartozik. Például az EV_REL különbséget tesz az X, az Y és a Z tengelyek, valamint a függőleges és vízszintes görgőtengelyek között. Hasonlóképpen az EV_KEY gombok és egérgombok százai között tesz különbséget.
Az eseményrétegen keresztül az EVIOCGBIT ioctl segítségével minden egyes eszköz tulajdonságai és képességei megadhatók. Ezzel a függvénnyel meghatározható, hogy egy eszköz milyen fajta tulajdonságokkal bír, például vannak-e billentyűi, gombjai, esetleg egyik sem. Ezen túlmenően az is meghatározható, hogy az eszköz pontosan milyen lehetőségeket támogat, például milyen billentyűk vagy gombok találhatók rajta.
Az EVIOCGBIT ioctl 4 kapcsolóval hívható meg, és a következőképpen néz ki:

```
ioctl(fd, EVIOCGBIT(ev_type, max_bytes), bitfield);
```

Az fd egy megnyitott fájlhoz tartozó fájlleírót jelöl, az

11. lista Az EVIOCGLED használata

```
memset(led_b, 0, sizeof(led_b));
ioctl(fd, EVIOCGLED(sizeof(led_b)), led_b);

for (yalv = 0; yalv < LED_MAX; yalv++) {
    if (test_bit(yalv, led_b)) {
        /* a bit be van kapcsolva */
        printf("  LED 0x%02x ", yalv);
        switch ( yalv)
            {
            case LED_NUML :
                printf(" (Num Lock)\n");
                break;
            case LED_CAPSL :
                printf(" (Caps Lock)\n");
                break;
            /* other LEDs not shown here*/
            default:
                printf(" (Unknown LED:
                ➥0x%04hx)\n",
                yalv);
            }
    }
}
```

ev_type a lekérdezendő tulajdonságok típusát adja meg (0-s típus esetén minden tulajdonságot lekérdezünk, nem csak az adott típushoz tartozókat). A max_bytes azt jelöli, hogy legfeljebb hány bájtnyi adattal térhet vissza a függvény, végül pedig a bitfield arra a területre mutat, ahova a lekérdezett adatokat a függvénnyel másoltatni szeretnénk. A függvény visszatérési értéke a bitfield területre ténylegesen átmásolt bájtok számát adja, vagy hiba esetén a hibakód negatív értékét.
Lássunk néhány EVIOCGBIT ioctl hívást tartalmazó példát. A 7. listában (49. CD Magazin/USB könyvtár) található kód például megmutatja, hogyan kérdezzük le az eszköz tulajdonságait. A kódban az evtype_bitmask méretét a *<linux/input.h>* fájlban található EV_MAX érték segítségével adjuk meg. Ezt követően kiadjuk az ioctl hívást, és az eseményréteg feltölti a bittömbünket. Ezután a tömb összes bitjét megvizsgáljuk, hogy lássuk, melyek vannak bekapcsolva, továbbá hogy az eszköz rendelkezik-e a tulajdonságok valamelyikével. A 2.5-ös rendszermagban minden eszköz támogatja az EV_SYN tulajdonságot, ezt a bitet a bemeneti mag kapcsolja be.
Ha a billentyűzet adatait kérdezzük le, akkor a következő eredményt kapjuk:

Támogatott eseménytípusok
- Eseménytípus 0x00 (összehangoló események)
- Eseménytípus 0x01 (billentyűk vagy gombok)
- Eseménytípus 0x11 (LED-ek)
- Eseménytípus 0x14 (ismétlés)

Az egér lekérdezése esetén a következőt láthatjuk:

Támogatott eseménytípusok
- Eseménytípus 0x00 (összehangoló események)
- Eseménytípus 0x01 (billentyűk vagy gombok)
- Eseménytípus 0x02 (viszonylagos tengelyek)

12. lista Az ismétlési beállítások lekérdezése: int rep[2];

```
if(ioctl(fd, EVIOCGREP, rep)) {
    perror("evdev ioctl");
}

printf("[0]= %d, [1] = %d\n", rep[0],
➥rep[1]);
```

13. lista Az ismétlés beállítása

```
int rep[2];

rep[0] = 2500;
rep[1] = 1000;

if(ioctl(fd, EVIOCSREP, rep)) {
    perror("evdev ioctl");
}
```

14. lista A letapogató kódok kiírása

```
int codes[2];

for (i=0; i<130; i++) {
    codes[0] = i;
    if(ioctl(fd, EVIOCGKEYCODE, codes)) {
        perror("evdev ioctl");
    }
    printf("[0]= %d, [1] = %d\n",
            codes[0], codes[1]);
}
```

15. lista A billentyűátrendezés

```
int codes[2];

codes[0] = 58; /* M keycap */
codes[1] = 49; /* assign to N */

if(ioctl(fd, EVIOCSKEYCODE, codes)) {
    perror("evdev ioctl");
}
```

Bemenet lekérdezése az eszköztől

Miután megállapítottuk, hogy egy eszköz milyen képességekkel rendelkezik, tudni fogjuk, hogy milyen eseményekre számíthatunk tőle és milyen eseményeket küldhetünk neki. Az események olvasásához csak egy egyszerű „olvasás” műveletet szükséges kezdeményezünk az eszközhöz tartozó karakteres eszközön. Minden alkalommal, ha az eseményeszközről olvasol (például */dev/input/event0*), egyszerre mindig események egész sorát fogod visszakapni, ahol minden esemény `input_event` szerkezetű.

A *8. listában* található kód egy ciklusban egy fájlleíróról olvas

17. lista A tengelyek mindenkori állapota

```
uint8_t abs_b[ABS_MAX/8 + 1];
struct input_absinfo abs_feat;

ioctl(fd, EVIOCGBIT(EV_ABS, sizeof(abs_b)),
➥abs_b);

printf("Supported Absolute axes:\n");

for (yalv = 0; yalv < ABS_MAX; yalv++) {
    if (test_bit(yalv, abs_b)) {
        printf("  Absolute axis 0x%02x ",
                ➥yalv);
        switch ( yalv)
            {
            case ABS_X :
                printf("(X Axis) ");
                break;
            case ABS_Y :
                printf("(Y Axis) ");
                break;
            default:
                printf("(Unknown abs
                ➥feature)");
            }
        if(ioctl(fd, EVIOCGABS(yalv),
&abs_feat)) {
            perror("evdev EVIOCGABS ioctl");
        }
        printf("%d (min:%d max:%d flat:%d
        ➥fuzz:%d)",
                abs_feat.value,
                abs_feat.minimum,
                abs_feat.maximum,
                abs_feat.flat,
                abs_feat.fuzz);
      printf("\n");
    }
}
```

be eseményeket. Kiszűr minden olyan eseményt, ami nem billentyűkhöz tartozik, majd kiírja az `input_event` szerkezet egyes elemeit. A programot futtatása közben a billentyűzeten gépeltem, ami a következő kimenetet adta:

```
Esemény: időpont 1033621164.003838, típus 1, kód 37, érték 1
Esemény: időpont 1033621164.027829, típus 1, kód 38, érték 0
Esemény: időpont 1033621164.139813, típus 1, kód 38, érték 1
Esemény: időpont 1033621164.147807, típus 1, kód 37, érték 0
Esemény: időpont 1033621164.259790, típus 1, kód 38, érték 0
Esemény: időpont 1033621164.283772, típus 1, kód 36, érték 1
Esemény: időpont 1033621164.419761, típus 1, kód 36, érték 0
Esemény: időpont 1033621164.691710, típus 1, kód 14, érték 1
Esemény: időpont 1033621164.795691, típus 1, kód 14, érték 0
```

Minden egyes billentyűlenyomáshoz és felengedéshez külön esemény tartozik. Az eseményréteg olvasása a karakteres eszközök olvasásának jellemzőivel bír, vagyis egy ciklusban nem kell folyamatosan kiolvasnod az értékeket, kizárólag akkor, ha a programodnak az adott eszköztől valamilyen

bemenetre van szüksége. Ezen túlmenően, ha egyszerre több eszköz bemenetére is kíváncsi vagy, használhatod a `poll` és `select` függvényeket, hogy lásd, melyik eszközön van feldolgozható adat.
Eseményt ugyanolyan egyszerűen küldhetünk az eszköznek, mint ahogyan fogadunk tőle, azzal a különbséggel, hogy a `read` függvény helyett a `write` függvényt kell meghívnunk egy `input_event` szerkezetű eseménnyel. Erre a *9. listában* láthatunk példát. A példaprogram bekapcsolja a CAPS LOCK kijelzőjét, vár 200 milliszekundumot, majd lekapcsolja a kijelzőt. Majd elvégzi ugyanezt a NUM LOCK-kal, miután újból megismétli az első műveletet (egy végtelen ciklusban), így a billentyűzeten a két kijelző folyamatos villogását fogod tapasztalni.
Mostanra már világossá válhatott, hogy eseményeket csak akkor kapsz, ha valami történt – lenyomtak vagy felengedtek egy billentyűt, mozgatták az egeret stb. Néhány eszköz esetében ismerni kell az eszköz állapotát, például tudni szeretnéd, hogy egy billentyűzeten mely LED-ek vannak bekapcsolva, és melyek kikapcsolva, akkor is, ha a hozzájuk tartozó változást jelző esemény még a program indulása előtt következett be. Az `EVIOCGKEY ioctl` pontosan erre szolgál: lekérdezhetjük vele a billentyűk és gombok mindenkori állapotát. Használatára a *10. listában* láthatunk példát. Az `ioctl` nagyon hasonlatos az `EVIOCGBIT(...,EV_KEY,...)` függvényhez, de ahelyett, hogy a tömbben az eszköz gombjainak vagy billentyűinek a listáját küldené el, az `EVIOCGKEY` csak a lenyomott gombokhoz vagy billentyűkhöz tartozó biteket állítja be.
Az `EVIOCGLED` és az `EVIOCGSND` függvények megegyeznek a `EVIOCGKEY`-vel, azzal a kivétellel, hogy a bekapcsolt LED-ek, illetve a bekapcsolt hangok listájával térnek vissza. Az `EVIOCGLED` használatára a *11. listában* láthatunk példát. Tehát még egyszer: az `EVIOCGBIT` által kitöltött tömbben minden bitet hasonló módon kell értelmezni.
Az `EVIOCGREP ioctl` hívással a billentyűzet ismétlési beállításait kérdezheted le. Ennek működésére a *12. listában* láthatsz példát, ahol a tömbben két érték található. Az első érték a késleltetést határozza meg, még mielőtt a billentyűzet elkezdené az ismételt küldést, míg a második érték az ismétlések közötti várakozási időt adja meg. Ennek megfelelően, ha lenyomsz egy billentyűt, rögtön kapsz egy karaktert, a következő karaktert `rep[0]` milliszekundum múlva kapod, az azt követőt `rep[1]` milliszekundum múlva, míg az azt követő összes többit `rep[1]` milliszekundumonként, egészen addig, amíg fel nem engeded a billentyűt.
Ezeket a beállításokat az `EVIOCSREP ioctl` hívással változtathatod meg. A *13. listában* látható, hogy ez a hívás ugyanazt a kételemű tömböt használja, amit a lekérdezésnél is használtunk. A példaprogram a kezdeti várakozási időt 2,5 másodpercben határozza meg, míg az ismétlési időt 1 másodpercben.
Néhány bemeneti eszközmeghajtó támogatja a lenyomott billentyűk leképezett visszaadását (ahogyan azokat a billentyűzet érzékeli, és letapogató kódokat – scancode – ad vissza), és az eseményeket úgy küldi el a bemeneti rétegnek. Az `EVIOCGKEYCODE ioctl` hívással eldöntheted, hogy az egyes kódokhoz milyen billentyűk tartoznak. A *14. listában* lévő program az első száz letapogató kódon lépdel végig egy ciklusban. A letapogató kód értéke (a függvény bemenete) az egészszámtömb első mezője, és az ahhoz tartozó gombszám-esemény (a billentyűkód) a tömb második eleme.
Ezt a leképezést az `EVIOCSKEYCODE ioctl` hívással módosíthatod. A műveletet a *15. lista* szemlélteti, ahol az `ioctl` az M billentyűt az N-nek megfelelően képezi le, így az M minden egyes lenyomásakor egy N betűt kapsz vissza. Fontos, hogy a billentyűkódokat átállító `ioctl` függvények nem minden billentyűzeten működnek – például az USB-s billentyűzetek eszközmeghajtója nem támogatja az ilyen leképezések megadását.
Az `EVIOCGABS` ioctl is állapotadatokat ad vissza. De ahelyett, hogy egy bitmezőn jelölné be az egyes állapotokat, egy `input_absinfo` szerkezetet ad meg a rögzített tengelyre. Ha szükséged van az eszköz mindenre kiterjedő állapotára, akkor a függvényt meg kell hívnod minden létező tengelyre, mint az a *17. listában* látható. A tömbben lévő értékek előjeles, 32 bites mennyiségek, és nyugodtan kezelheted őket az `int32_t`-vel megegyező típusúként. Az első elem a tengely jelenlegi értékét adja meg, míg a második és a harmadik elemek a tengely jelenlegi határait jelöli ki, a 4. elem a válasz „flat" területének méretét szolgáltatja (ha van ilyen), míg az utolsó érték hiba esetén a hibához tartozó terület méretét adja meg.

Erő-visszacsatolás

Három további függvény létezik az erő-visszacsatolásos eszközök kezelésére: az `EVIOCSFF`, az `EVIOCRMFF` és az `EVIOCGEFFECT`. Ezek a függvények erővisszacsatolás-hatásokat küldenek, vonnak vissza, illetve megadják, hogy egyidejűleg hány hatás alkalmazható (ebben a sorrendben). Mivel az erő-visszacsatoláshoz tartozó programozói felület még fejlődik, illetve változásban van, egyelőre még korai lenne a teljes programozói felületet ismertetni. A *Kapcsolódó címek* rész tartalmaz olyan helyeket, amelyek talán már e cikk megjelenésekor is frissített adatokkal rendelkeznek a témakörben.

A cikkhez tartozó listák megtalálhatóak a 49. CD Magazin/USB könyvtárában.

Linux Journal 2003. március, 107. szám

Brad Hards
(bradh@frogmounth.net) A Sigma Bravo technikai igazgatója egy szakértői szolgáltatásokat nyújtó kis cégnél, Canberrában. A Linux mellett repülőgép-rendszerek összeillesztésével és minősítésével is foglalkoznak.

KAPCSOLÓDÓ CÍMEK

A Linux bemeneti alrendszer elsősorban a 2.5-ös BitKeeper rendszermagban található meg.
A BitKeeper nagyon sokoldalú rendszer, de ha pusztán a rendszermagot szeretnéd böngészni, a
➲ http://linus.bkbits.net:8080/linux-2.5 cím hasznos kiindulópontként szolgálhat.
Létezik egy kísérleti fejlesztési fa kizárólag a bemeneti alrendszerhez, amit elérhetsz a
➲ http://linux-input.bkbits.net:8080/linux-input címen.
Korábban a bemeneti alrendszer a
➲ http://linuxconsole.sourceforge.net címen volt megtalálható, de a rendszermaggal együtt átköltözött a BitKeeper alá. Bár az oldal nem tartalmaz túl sok dokumentációt, található rajta néhány hasznos folt felhasználói szintű alkalmazásokhoz.

Linuxos jelzések alkalmazásfejlesztői szemszögből

A jelzések alapvető eszközt jelentenek a folyamatok közötti adatcserében, és a hálózati kiszolgálóktól kezdve a médialejátszókig gyakorlatilag mindenben alkalmazzák őket. Tanuld meg te is, hogyan használhatsz jelzéseket a saját programjaidban!

A jelzések működésének pontos megértése elengedhetetlen a linuxos környezetben dolgozó alkalmazáskészítők számára. A jelzéskezelés, illetve a vele kapcsolatos függvények ismerete révén a fejlesztők hatékonyabban végezhetik a munkájukat.

Az alkalmazások futtatása – ha minden utasítás rendben lefut – lépések egymásutánját jelenti. Ha hiba vagy rendellenesség történik egy program futása közben, a rendszermag jelzések segítségével értesítheti erről a folyamatot. A jelzéseket régebben a folyamatok közötti adatcserére és összehangolásra, valamint a folyamatok közötti adatcserék (IPC) egyszerűsítésére is használták. Noha ma már sokkal fejlettebb összehangoló eszközök és IPC-módszerek is rendelkezésünkre állnak, Linux alatt a jelzések továbbra is rendkívül fontos szerepet játszanak a kivételek és megszakítások kezelésében. A jelzéseket körülbelül harminc éven át lényegesebb módosítás nélkül használták. Az első 31 jelzés szabványos, ezek némelyike még az 1970-es évekből, a Bell Laboratórium Unix fejlesztéseiből származik. A Posix (Portable Operating Systems and Interface for UNIX) szabvány új, valós idejű jelzésosztályt vezetett be, amelynek tagjai 32-től 63-ig számozódnak.

Ha hiba történik, a rendszer jelzést hoz létre, majd az eseményt a rendszermag átadja a fogadó folyamatnak. Egyes esetekben jelzést folyamat is küldhet egy másik folyamatnak. A folyamat-folyamatjelzések mellett számos olyan helyzet van, amikor a jelzés a rendszermagtól ered: ilyen például, ha egy fájl mérete meghaladja az előírt határt, ha egy be-, illetve kiviteli eszköz készen áll, ha a rendszer érvénytelen utasítást hajtott végre, vagy ha egy terminál Ctrl-C vagy Ctrl-Z megszakítást küld. Minden jelzés neve SIG-gel kezdődik, és egyedi, pozitív egész számként van megadva. A héj parancssorában a `kill -l` parancsot kiadva az összes jelzés nevét és sorszámát megjeleníthetjük. A jelzések számai a */usr/include/bits/signum.h* állományban vannak megadva, ennek forrása a */usr/src/linux/kernel/signal.c*. Egy folyamat akkor kap jelzést, ha felhasználói módban fut. Ha a fogadó folyamat magmódban fut, akkor a jelzés végrehajtása csak akkor kezdődik meg, ha a folyamat újra felhasználói módba vált át.

A nem futó folyamatoknak küldött jelzéseket a rendszermagnak kell mentenie, amíg az adott folyamat futtatása újra nem indul. Az alvó folyamatok lehetnek megszakíthatók és nem megszakíthatók. Ha egy folyamat megszakítható alvó állapot mellett kap jelzést (például éppen terminál be-, illetve kivitelre vár), akkor a rendszermag felébreszti, hogy a folyamat kezelhesse a jelzést. Ha egy folyamat nem megszakítható állapotban kap jelzést (például lemezműveletre vár), a rendszermag a művelet befejeződéséig visszatartja a jelzést.

Ha egy folyamat jelzést kap, három dolog történhet. Az első lehetőség, hogy a folyamat figyelmen kívül hagyja a jelzést. A második, hogy elfogja a jelzést, és egy különleges, jelzéskezelőnek nevezett függvényt hív meg. A harmadik lehetőség az, hogy végrehajtja a jelzéshez rendelt alapértelmezett műveletet. A 15-ös számú `SIGTERM` jelzéshez például a folyamat befejezése tartozik mint alapértelmezett művelet. Egyes jelzések nem hagyhatók figyelmen kívül, másokhoz viszont nem tartoznak alapértelmezett műveletek, így ez utóbbiaknál a figyelmen kívül hagyás az alapértelmezett művelet. A jelzésnevek, -azonosítók és az alapértelmezett műveletek listáját, illetve azt, hogy mely jelzések foghatók el, a signal(7) súgóoldalon találhatod meg.

Ha újabb jelzés érkezik, miközben egy folyamat egy jelzéskezelőt futtat, az újabb jelzést a rendszer addig visszatartja, amíg a jelzéskezelő futása véget nem ér. A cikk további részei a jelzésekkel kapcsolatos alapokat, függvényeket, valamint alkalmazásukat tárgyalják.

Jelzések a rendszermagon belül

Vajon hol tároljuk a jelzésekkel kapcsolatos adatokat a folyamatban? A rendszermag egy kötött méretű, `proc` adatszerkezetekből álló tömbbel rendelkezik, ez a folyamattábla. A `proc` adatszerkezetek `u` (user, felhasználói) területe tárolja a folyamatokkal kapcsolatos vezérlőadatokat. Az `u` terület fontosabb mezői tartalmazzák – többek között – a jelzéskezelőket és a kapcsolódó adatokat. A jelzéskezelő egy tömb, elemei a rendszerben megadott jelzésekhez tartoznak, és az adott folyamat által az adott jelzés fogadásakor végrehajtott műveletet jelölik ki. A `proc` adatszerkezet jelzéskezelési adatokat tart fenn, lényegében a figyelmen kívül hagyandó, gátolandó, átadandó és kezelendő jelzések maszkjait.

Miután a rendszer jelzést hozott létre, a rendszermag egy bitet állít be a folyamattábla bejegyzésének jelzésmezőjében. Ha a jelzést figyelmen kívül hagyjuk, a rendszermag további műveleteket már nem hajt végre. Mivel a jelzésmező jelzésenként egy bitet jelent, ugyanannak a jelzésnek többszöri előfordulását a rendszer nem tartja nyilván.

A jelzés átadásakor a fogadó folyamatnak a jelzéstől függően kell cselekednie. A végrehajtott művelet lehet a folyamat befejezése, memóriakiíratás és a folyamat befejezése, a jelzés figyelmen kívül hagyása, a felhasználó által megadott jelzéskezelő meghívása (ha a folyamat elfogja a jelzést) vagy a folyamat futtatásának folytatása, ha ideiglenesen fel volt függesztve.

A memóriakiíratás egy *core* nevű fájlba történik, ami a befejeződött folyamat képét tartalmazza. Megtalálni benne a folyamat változóit, illetve a veremnek a hiba pillanatában fennállt állapotát. A *core* fájlból a programozó egy hibakeresővel kinyomozhatja a folyamat befejeződésének okát. A core (mag) megnevezés itt történelmi okokra vezethető vissza: régen a központi memória fánk alakú, induktormagoknak nevezett mágnesekből állt. Egy jelzés elfogása azt jelenti, hogy jelezzük a rendszermagnak: ha valamilyen jelzés lép fel, akkor az alapértelmezett helyett a program saját jelzéskezelőjét kell futtatnia. Két kivétel van: a `SIGKILL` és a `SIGSTOP`, ezeket nem lehet elfogni vagy figyelmen kívül hagyni.

1. lista Jelzés elfogása és figyelmen kívül hagyása

```
#include <signal.h>

void saját_kezelő (int sig);
/* a függvény prototípusa */

int main ( void ) {

/* 1. rész: SIGINT elfogása */
     signal (SIGINT, saját_kezelő);
     printf ("SIGINT elfogása\n");
     sleep(3);
     printf ("3 másodpercen belül nem
              ➥érkezett SIGINT\n");

/* 2. rész: SIGINT figyelmen kívül hagyása */
     signal (SIGINT, SIG_IGN);
     printf ("SIGINT figyelmen kívül
              ➥hagyása\n");
     sleep(3);
     printf ("3 másodpercen belül nem
              ➥érkezett SIGINT\n");

/* 3. rész: Alapértelmezett művelet SIGINT-
hez */
     signal (SIGINT, SIG_DFL);
     printf ("Alapértelmezett művelet
              ➥SIGINT-hez\n");
     sleep(3);
     printf ("3 másodpercen belül nem
              ➥érkezett SIGINT\n");
     return 0;
}

/* Felhasználó által megadott jelzéskezelő
   függvény */
void saját_kezelő (int sig) {
     printf ("SIGINT érkezett, azonosító:
              ➥%d\n", sig);
     exit(0);
}
```

A `sigset_t` egy alap-adatszerkezet a jelzések tárolására. A folyamatoknak küldött `sigset_t` adatszerkezet egy olyan bittömb, amiben minden jelzéstípushoz egy-egy bit tartozik:

```
typedef struct {
                unsigned long sig[2];
             }  sigset_t;
```

Mivel minden előjel nélküli `long` szám 32 bites, Linux alatt – a Posix-szabványnak megfelelően – legfeljebb 64 jelzés adható meg. A nullához nem tartozik jelzés, így a `sigset_t` első felének további 31 eleme az első szabványos 31 jelzéshez tartozik, a második rész bitjei pedig a 32–64. sorszámú, valós idejű jelzésekhez rendelődnek. A `sigset_t` mérete 128 bájt.

A jelzések kezelése

Számos rendszerhívás és jelzéstámogatott könyvtári függvény létezik, amelyek segítségével egy-egy folyamaton belül könnyedén megoldható a jelzések kezelése. Elsőként a jó öreg `signal` rendszerhívással ismerkedünk meg, majd további hasznos függvényekről is szó esik, mint a `sigaction`, `sigaddset`, `sigemptyset`, `sigdelset`, `sigismember` és a `kill`.

2. lista Megegyezik az elsővel, de sigaction hívásokat használ

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

void saját_kezelő (int sig);
/* a függvény prototípusa */

int main ( void ) {

     struct sigaction saját_művelet;

/* 1. rész: SIGINT elfogása */

     saját_művelet.sa_handler = saját_kezelő;
     saját_művelet.sa_flags = SA_RESTART;
     sigaction (SIGINT, &saját_művelet, NULL);
     printf ("SIGINT elfogása\n");
     sleep(3);
     printf ("3 másodpercen belül nem
              ➥érkezett SIGINT\n");


/* 2. rész: SIGINT figyelmen kívül hagyása */

    saját_művelet.sa_handler = SIG_IGN;
    saját_művelet.sa_flags = SA_RESTART;
    sigaction (SIGINT, &saját_művelet, NULL);
    printf ("SIGINT figyelmen kívül
             ➥hagyása\n");
    sleep(3);
    printf (" Az alvó állapot (sleep)
             ➥véget ért \n");


/* 3. rész: alapértelmezett művelet SIGINT-hez */

   saját_művelet.sa_handler = SIG_DFL;
   saját_művelet.sa_flags = SA_RESTART;
   sigaction (SIGINT, &saját_művelet, NULL);
   sleep(3);
   printf ("3 másodpercen belül nem érkezett
            ➥SIGINT\n");
}

void saját_kezelő (int sig) {
     printf ("SIGINT érkezett, azonosító:
              ➥%d\n", sig);
     exit(0);
}
```

A signal rendszerhívás

A `signal` rendszerhívást az adott jelzés elfogására, figyelmen kívül hagyására, valamint a hozzárendelt alapértelmezett művelet megadására használjuk. Két átadott értéket vár: a jelzés számát és egy mutatót a felhasználó által megadott jelzéskeze-

3. lista Az SA_SIGINFO és az sa_sigaction használata adatok kinyerésére egy jelzésből

```
#include <unistd.h>
#include <sys/types.h>
#include <signal.h>
#include <bits/siginfo.h>
#include <stdio.h>

void kezelő (int signo, siginfo_t *info,
             ➥void *context);

main () {

   struct sigaction saját_művelet;

   saját_művelet.sa_flags = SA_SIGINFO;
   saját_művelet.sa_sigaction = kezelő;

   sigaction (SIGINT, &saját_művelet, NULL);

   printf ("SIGINT elfogása\n");
   sleep(5);
   printf ("Kész.\n");
}

void kezelő (int signo, siginfo_t *info,
             ➥void *context)
 {
    printf ("Jelzés azonosítója: %d\n",
             ➥info->si_signo);

 /* A siginfo_t adatszerkezet elemeit a
    man 2 sigaction ismerteti. */
}
```

4. lista SIGINT-et fogadó és küldő programok

```
#include <signal.h>

main ( ) {
    int folyamat_azonosító;
    printf ("Add meg annak a folyamatnak
             ➥az azonosítóját amelynek a
             ➥jelzést akarod küldeni : ");
    scanf ("%d", &folyamat_azonosító);

   if (!(kill ( folyamat_azonosító, SIGINT)))
       printf ("SIGINT elküldve a következő
                ➥azonosítójú folyamatnak:
                ➥%d\n", folyamat_azonosító);
   else if (errno == EPERM)
       printf ("A művelet nincs
                ➥engedélyezve.\n");
   else
       printf ("%d nem létezik \n",
                ➥folyamat_azonosító);
}

/* 4a kódrészlet Ez a program addig fut,
   amíg SIGINT-et nem kap. */

#include <signal.h>

 main ( ) {
   printf (" Ennek a folyamatnak az
            ➥azonosítója: %d. "
   " SIGINT várása.\n", getpid());
   for (;;);
}
```

lőre. Linux alatt két fenntartott, előre megadott jelzéskezelő áll rendelkezésünkre, a `SIG_IGN` és a `SIG_DFL`. A `SIG_IGN` a megadott jelzést figyelmen kívül hagyja, a `SIG_DFL` pedig az adott jelzésre vonatkozóan az alapértelmezettet állítja be jelzéskezelőnek (lásd: `man 2 signal`).
Siker esetén a rendszerhívás az adott jelzés jelzéskezelőjének korábbi értékével tér vissza. Ha a `signal` hívás sikertelen, visszatérési értéke `SIG_ERR`. Az *1. lista* 1. kódrészletében a `SIGINT` elfogásának, figyelmen kívül hagyásának és alapértelmezett műveletének megadása látható. Minden résznél nyomd le a CTRL-C billentyűkombinációt, ez `SIGINT` jelzést vált ki.

sigaction

A `sigaction` rendszerhívás a `signal` helyett használható; szélesebb körű vezérlési lehetőségeket ad a jelzésekhez. Írásmódja a következő:

```
int sigaction ( int signum, const struct
                ➥sigaction *act,
                ➥struct sigaction *oldact);
```

Első átadott értéke (`signum`) egy adott jelzést jelöl, a második (`sigaction`) a `signum` jelzés új kezelőjének megadására szolgál, a harmadik pedig a korábbi kezelő – ez általában `NULL` – tárolását végzi.

A `sigaction` adatszerkezet felépítése a következő:

```
struct sigaction {
    void (*sa_handler)(int);
    void (*sa_sigaction)(int, siginfo_t *,
                         ➥void *);
    sigset_t sa_mask;
    int sa_flags;
}
```

A sigaction adatszerkezet tagjainak a leírása:

- `sa_hander` – mutató egy felhasználó által vagy előre megadott (`SIG_IGN` vagy `SIG_DFL`) jelzéskezelőre.
- `sa_maskn` – jelzések egy maszkját adja meg a jelzés kezelésekor. A jelzések gátlásának elkerülésére a `SA_NODEFER` vagy a `SA_NOMASK` jelző használható.
- `sa_flags` – jelzéshez tartozó műveletet ad meg. Több jelző is rendelkezésre áll a jelzés különféle módokon való kezelésére. Egynél több jelző `OR` művelettel használható:
  - `SA_NOCLDSTOP` – amennyiben megadjuk a `SIGCHLD` jelzést, ha a gyermek futása véget ért, nem kap jelzést.
  - `SA_ONESHOT` vagy `SA_RESETHAND` – a felhasználó által megadott jelzéskezelő végrehajtása után visszaállítja a jelzéshez tartozó alapértelmezett műveletet. Az alapértelmezett művelet beállításának megakadályozására az `SA_RESTART` használható.

- SA_NOMASK vagy SA_NODEFER – a jelzés maszkolását akadályozza meg. Az SA_SIGINFO a jelzéssel kapcsolatos adatok megszerzésére alkalmas.
- sa_sigaction – ha az SA_SIGINFO jelzőt használjuk az sa_flags-ben ahelyett, hogy a jelzéskezelőt az sa_handler-ben adnánk meg, akkor az sa_sigaction-t kell alkalmazni.

Az sa_sigaction mutató egy függvényre, ami három átadott értéket vár – ellentétben az sa_handler-rel, ami csak egyet –, például:

```
void saját_kezelő
➥(int signo, siginfo_t info, *void *context)
```

A signo ebben az esetben a jelzés száma, az info pedig egy mutató egy siginfo_t típusú adatszerkezetre, ami a jelzéssel kapcsolatos adatokat tartalmazza. A context szintén mutató egy ucontext_t típusú objektumra, ami arra a fogadó folyamat-összefüggésre hivatkozik, amit a jelzés megszakított.
A második kódrészlet hasonló az elsőhöz, de a signal helyett a sigaction rendszerhívást használja. A *3. lista* a jelzésekkel kapcsolatos adatokat ismerteti a SIG_INFO jelző segítségével.

Jelzések küldése

Eddig a CTRL-C billentyűkombinációval küldtünk SIGINT jelzést a héjból. Ha ezt programból szeretnénk megtenni, a kill rendszerhívást kell használnunk, aminek két értéket kell átadnunk: a folyamat azonosítóját és a jelzés számát.

```
int kill ( pid_t folyamat_azonosító,
           ➥int jelzés_száma );
```

Ha a folyamat azonosítója pozitív, a jelzést egy megadott folyamat kapja. Ha negatív, a jelzést a rendszer annak a folyamatnak küldi, amelynek a csoportazonosítója megegyezik a folyamatazonosító abszolút értékével.
Talán nem meglepő, hogy az önálló programként (/bin/kill) is létező, illetve a bash részeként is használható kill parancs (próbáld ki: help kill) a kill rendszerhívással küld jelzéseket. Nem minden folyamat küldhet jelzést a többinek. Ahhoz, hogy egy folyamat jelzést küldhessen egy másiknak, a küldőnek rendszergazdaként kell futtatnia, vagy a küldő valódi (effective) felhasználóazonosítójának ugyanannak kell lennie, mint a fogadó folyamat valódi vagy mentett azonosítójának. Ez azt jelenti, hogy a te héjad, ami a te jogosultságaiddal fut, jelzést tud küldeni egy általad indított, de éppen rendszergazdai módban futó setuid programnak:

```
# cp /bin/sleep ~/rootsleep
# sudo chmod u+s ~/rootsleep
# ./rootsleep 40
# killall rootsleep
# rm ~/rootsleep
```

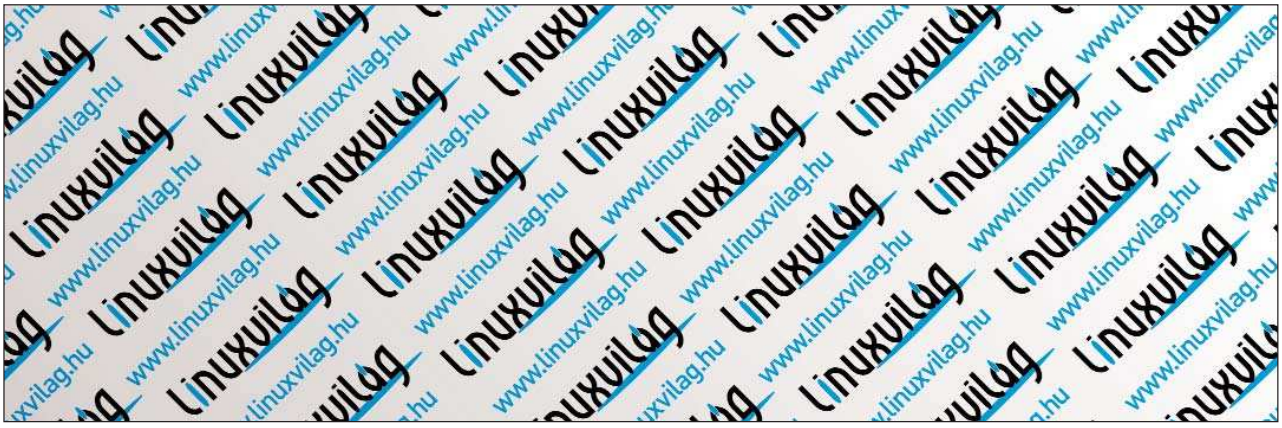
Egy egyszerű felhasználó nem tud jelzést küldeni a rendszerfolyamatnak, például a swapper és az init.
A kill annak a megállapítására is alkalmas, hogy egy folyamat létezik-e vagy sem. Jelzésazonosítónak válaszd a 0-t; ha a folyamat létezik, a kill visszatérési értéke nulla, ellenkező esetben pedig -1 lesz.
A *4. lista* ***4a*** jelű kódrészlete a kill rendszerhívás használatát szemlélteti. Először indítsd el a 4a programot, és nézd meg a folyamatazonosítóját. Ezután – egy másik ablakban – indítsd el a 4. lista programját, és bemenetként a 4a program folyamatazonosítóját használd.

Linux Journal 2003. március, 107. szám

Dr. B. Thangaraju
Fizikából szerzett PhD fokozatot. Jelenleg a Wipro Technologies egyik vezetője, ugyancsak Indiában. Tekintélyes nemzetközi folyóiratokban több kutatási témájú írása is megjelent.

KAPCSOLÓDÓ CÍMEK

W. Richard Stevens Advanced Programming in the UNIX Environment, Pearson Education Asia, 1993, 263–324. oldal
Michael K. Johnson és *Erik W. Troan* Linux Application Development, Addison-Wesley, 1998
Moshe Bar „The Linux Signals Handling Model", Linux Journal, 2000. május 1. (elérhető a ➲ http://www.linuxjournal.com/article/3985 címen).
D. P. Bovet és *M. Cesati* Understanding the Linux Kernel, O'Reilly & Associates, 1998, 233–248 oldal



Bevezetés az Emacs használatába

Az Emacs a mindennel felszerelt szövegszerkesztő. Tanuljuk meg az alapjait, a végén talán még a határidőnaplónkat is ebben fogjuk tartani.

Ecikk gyorstalpaló Emacs-tanfolyam szeretne lenni. Feltételezzük, hogy az Emacs (ejtsd: imex) már telepítve van és fut (a legtöbb Linux-terjesztésnél ez nem gond). Az Emacs beépített oktatóprogramját is ismertnek vesszük. Az Emacs segítségével történő programfejlesztés nem témája ennek a cikknek, mert már írtam róla a Linuxvilág 2002. júniusi számában.

Az Emacs indításához gépeljük be az `emacs &` parancsot. Az `&` jel az Emacset a háttérbe teszi; s mivel az X jeleníti meg, ez a beállítás megfelelő.

Az Emacs a konzolon is futtatható, ekkor `&` nélkül adjuk ki a parancsot. Ha egy `xterm` ablakban szeretnénk futtatni, új ablak megnyitása nélkül, az `emacs -nw` parancsot kell használnunk. A konzolos és az xtermes üzemmód nagyon hasznos, ha nincs X, például amikor egy távoli kiszolgálón SSH-kapcsolaton keresztül dolgozunk. Az SSH beállítható az X továbbítására, és ha ezt engedélyezzük, akkor futtathatjuk a távoli Emacset (és más X-alkalmazásokat).

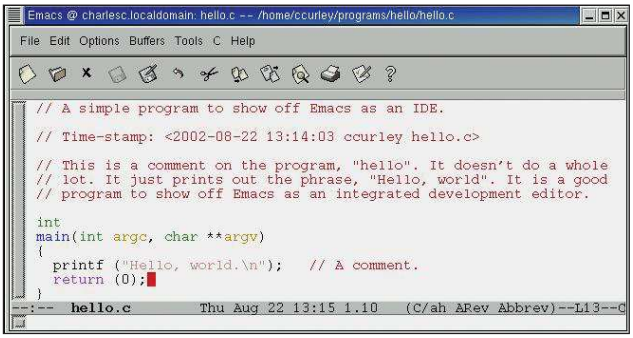
Ha még nem jutottunk el a beépített Emacs-tanfolyam végére, most tegyük meg. Nyomjuk meg a CTRL-H billentyűkombinációt, majd a T billentyűt. Az oktatóprogram a számítástechnika őskorában született (1985), így nem veszi figyelembe a kurzorvezérlő billentyűket és más korszerű kényelmi szolgáltatásokat. Az Emacs támogatja ezeket, de az oktatóprogram nem veszi figyelembe. Ha nem is tanuljuk meg, nem árt tudunk bizonyos Emacs-gyorsbillentyűk létezéséről, a bash és még sok GNU-program is beállítható a használatukra. Például a CTRL-B és a CTRL-N billentyűkombinációknak pontosan ugyanaz a hatása az Emacsben, mint a bashban. Valójában az Emacs stílusú gyorsbillentyűk az alapértelmezettek a bashben.

Az oktatóprogram segítségével elsajátíthatjuk az alapvető kurzormozgatási ismereteket, az Emacs-parancsok megszakítását, az Emacs ablakkezelését, a szövegtárolók (bufferek) és a fájlok kapcsolatát stb. Talán azt a legszükségesebb megjegyeznünk ebből, hogy a mozgatóbillentyűk általában adott billentyűk (az F előre), és a mozgatás tartományát a módosítók határozzák meg. Például a CTRL-F a kurzort egy betűhellyel mozgatja előre, a M-F viszont egy szóval előbbre helyezi (a M az Emacs jelölése a Meta billentyűre, ami a legtöbb billentyűzeten ALT-ként található meg).

Az Emacs jóval a webböngészők megjelenése előtt létezett már, ezért a „keret” kifejezést az X-es ablakra, az „ablak” kifejezést pedig a kereten belüli szakaszra használja. Mivel ez a cikk az Emacsről szól, az Emacs szóhasználatát fogjuk alkalmazni.

A CTRL-X 2 billentyűkombináció hatására egy új ablak jelenik meg, vízszintes elválasztással. A függőleges elválasztású új ablakhoz a CTRL-X 3 billentyűkombinációt kell alkalmaznunk. A CTRL-X 5 új keretet nyit, a CTRL-X 0 és a CTRL-X 5 0> bezárja a pillanatnyi ablakot, illetve keretet. Az oktatóprogram másik kiemelt része az Emacs növekményes keresőparancsainak ismertetése. Nagyon megkönnyítik az életet, ezért tanuljuk meg jól őket!

Az Emacs súgórendszere a CTRL-H billentyűkombinációval hívható elő. A CTRL-H? hatására a súgórendszer különböző részeit felsoroló lista jelenik meg. Az inforendszer (CTRL-H I) az FSF infoformátumában elérhető dokumentumok böngészését teszi lehetővé. Ez egy faszerkezetbe rendezett hiper-



1. kép Az Emacs C-módja a nyelvi elemek színezésével. A behúzásokról a C-mód gondoskodott

szöveges rendszer, még a World Wide Web előtti időkből.

Az Emacs szolgáltatásainak Info-fejezeteihez a CTRL-H CTRL-F billentyűkombinációval juthatunk el. Ez a fejezet a pillanatnyi fő- és almódról (részletesen lásd később), az Emacsre vonatkozó felhasználási szerződésről és garanciáról jelenít meg adatokat. Az Emacs súgóját az Emacs jeleníti meg, ezért az oktatóprogramban megtanult kurzormozgató billentyűkombinációk a súgóban is használhatóak.

Ismerkedés a módokkal

Az Emacs legelső súgóoldala szerint az Emacs bővíthető, testreszabható, önleíró, valós idejű megjelenítéssel dolgozó szövegszerkesztő. Bővíthető, mert Emacs Lisp nyelven, más néven Elisp nyelven írták, ami a Lisp nyelvnek kimondottan az Emacs és a szövegfeldolgozás számára testreszabott változata. A Emacs tudását Elisp nyelvű kód írásával bővíthetjük. Ezenkívül az Emacs viselkedése a meglévő elisp-változók értékeinek megváltoztatásával testreszabható. Az önleíró jelző talán kicsit túlzás, de az Elisp tényleg támogatja a programozót a kódja dokumentálásában. Láttuk már, hogy részletes súgó áll a rendelkezésünkre.

Az Emacset a felhasználó is testreszabhatja egy adott feladat elvégzésére, ezt az úgynevezett főmód váltásával érjük el. Egy szövegtárolóban egyszerre csak egy főmód lehet működő, de menet közben is lehet főmódot váltani. Például CGI-programok írásánál hasznos lehet a Perl-mód és a HTML Helper-mód közti váltogatás.

A szövegtárolóban pillanatnyilag érvényes mód azonosításához vessünk egy pillantást a módsorra. Zárójelben található egy vagy több mód, és a pillanatnyi főmód az első a sorban. Nem minden almód azonosítja magát a módzárójelekben, de a tevékenységük magától értetődik, például a ***Column Number***-(oszlopszám) mód.

Főmódok

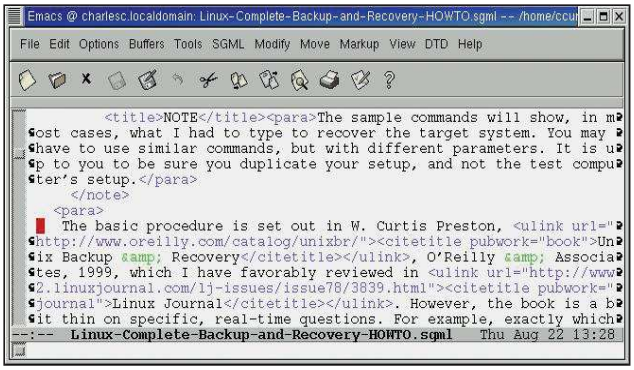
A főmódok általában az állományok kiterjesztéséhez vannak kötve. Az `auto-mode-alist` Lisp-változó gondoskodik a hozzárendelésekről, és meg fogjuk mutatni, hogyan adhatók hozzá további hozzárendelések. Az Emacs a parancsfájlok első sorának különleges bejegyzéséből is képes meghatározni az állomány típusát, például a következőhöz a Perl-módot rendeli:

```
#! /usr/bin/perl
```

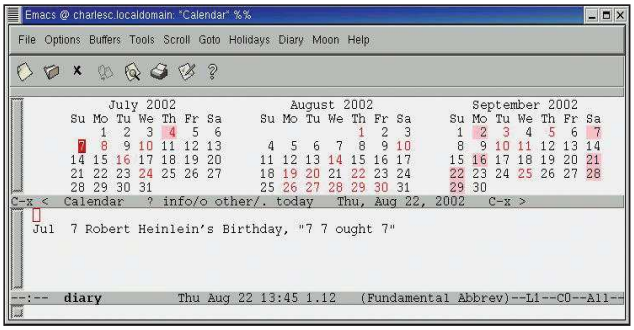
A módot mindig kikényszeríthetjük, ha a dokumentum első sorába a kívánt mód nevét `-*-` jelek közé tesszük:

```
# -*- shell-script -*-
```

Kézzel a `M-X mód neve` paranccsal válthatunk egyik módról a másikra, például a `M-X perl-mode` az Emacset a Perl-programok szerkesztéséhez használható főmódba váltja át.
A főmódok számos hasznos szolgáltatást nyújtanak. Általában az adott állománytípusra jellemző behúzási szabályokat és áttekinthető megjelenítést tesznek lehetővé. Sokszor lehetőség nyílik a megjegyzések beszúrására egy rövid billentyűkombináció segítségével. Egy szövegterület egyszerű mozdulattal helyezhető megjegyzésbe: az `M-X comment-region` paranccsal. Ha minden szerkesztenivalónkat az Emacsszel végezzük, az többek között azért hasznos, mert az egyik főmódban megismert szolgáltatás (valamint a gyorsbillentyűje és menüparancsa) nagy valószínűséggel megvan a többiben is, azaz ha képesek vagyunk a C-fájlokat szerkeszteni az Emacsben, akkor valószínűleg az SQL-fájlok szerkesztése is menni fog (*1. kép*).
A főmódok általában színekkel emelik ki a nyelvi elemeket. A nyelvi elemek és a színek egymáshoz rendelése önműködően történik. Például a megjegyzések pirosak, az adattípusok zöldek és a karakterláncok világos pirosak. Az Emacsben történő szerkesztésnek az a másik előnye, hogy a színek hozzárendelése minden módnál ugyanaz, azaz a megjegyzések pirosak – akár assembly kódot, akár XML-t szerkesztünk.
A főmódok módosítják a billentyűk működését, általában a TAB és a DELETE billentyűét. A főmódok módfüggő parancsait a CTRL-C előtag vezeti be. Például a PSGML-módban a CTRL-C CTRL-V paranccsal lehet az SGML-dokumentumot érvényesíteni.
Az egyik legnagyobb tudású főmód ***Lennart Staflin*** PSGML-módja (lásd a ***Kapcsolódó címeket***). Lehetővé teszi SGML- és XML-elemek beszúrását, és önműködően a C-móddal összemérhető áttekinthető megjelenést ad a dokumentumnak.
A PSGML is színezi a nyelvi elemeket és egyéb jó tulajdonságai is vannak, de ezeken kívül beolvassa a DTD-t is, és kikényszeríti az elemek helyes egymásba ágyazását. Például a DocBookban nem engedi meg egy `<sect4>`-nak közvetlenül egy `<sect1>` környezetbe történő beszúrását. A PSGML egyúttal az előtétprogram az érvényesítőhöz (*2. kép*).
A következő főmódok lehetnek még hasznosak szinte bárki számára: Dired-mód, Ediff-mód, W3, a Calendar (naptár) és Diary (határidőnapló). A Dired-mód a könyvtárak szerkesztésére való. Könyvtárról könyvtárra ugrálhatunk, rámehetünk fájlokra és szerkeszthetjük a fájlok metaadatait, például a hozzáférési jogokat és a tulajdonost. A Dired-mód egyik legjobb tulajdonsága, hogy képes több fájlban keresni, és megjelölni a találatokat. A találatokra sorban rámehetünk és szerkeszthetjük őket. Így lehetséges a fájlok „tömeges” kezelése, például az átnevezésük vagy a törlésük.



2. kép A Linux Documentation Project egy dokumentumának szerkesztése a DocBook SGML DTD-vel Emacsben. Az elemek és az entitások színekkel vannak kiemelve

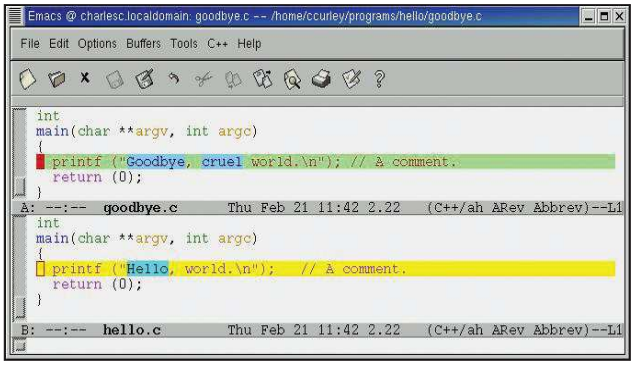


3. kép Az Emacs naptára és határidőnaplója

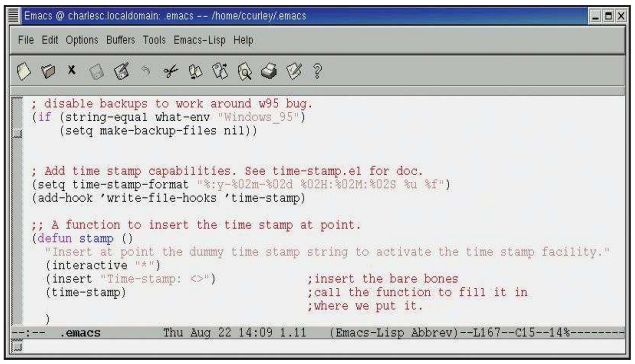
A naptár, illetve a határidőnapló meglehetősen hasznos mód. Az átlagos naptárprogramoktól eltérően a Calendar-mód lehetővé teszi a dátumok kezelését és átváltását a Gergely-naptár, a Juliánus-naptár, a kopt, a héber és az iszlám naptár között. Ha valami különlegességre vágyunk, a legközelebbi feljegyzésünket dátumozzuk a perzsa vagy a maja naptár szerint. Esetleg a következő hibajelentésünket a francia forradalmi naptár szerint küldjük el a Free Software Foundationnek (*3. kép*).
A jelentéktelen naptáraknál hasznosabb a határidőnapló, amiben találkozókat, évforduló-emlékeztetőket, ismétlődő eseményeket (például „minden hónap harmadik csütörtökje”) és más eseménytípusokat jegyezhetünk be. Ha megadjuk az esemény idejét, az Emacs figyelmeztet az idő közeledtekor. A határidőnapló nem pusztán önmagában hasznos, hanem mivel Emacsben fut, minden olyan számítógépen működik, amire létezik Emacs – márpedig a legtöbb számítógépre van. A naplóállomány szintén hordozható.
Az Ediff-mód foltok kiválasztott alkalmazására használható. Arra is jó, hogy több számítógépen frissítsük az állományokat, például a ***.emacs***- és a naplóállományokat. Mivel kiválasztható, az Ediff lehetővé teszi a változtatások mindkét irányú átvitelét. Ez fontos lehet, ha találkozókat állítunk be hordozható gépünkön, miközben a titkárnő szintén találkozókat állít be az asztali gépünkön (*4. kép*).
Ha a világhálót szeretnénk böngészni, vizsgáljuk meg ***William M. Perry*** W3-módját – ez egy Emacs Lisp nyelven írott webböngésző.

Almódok

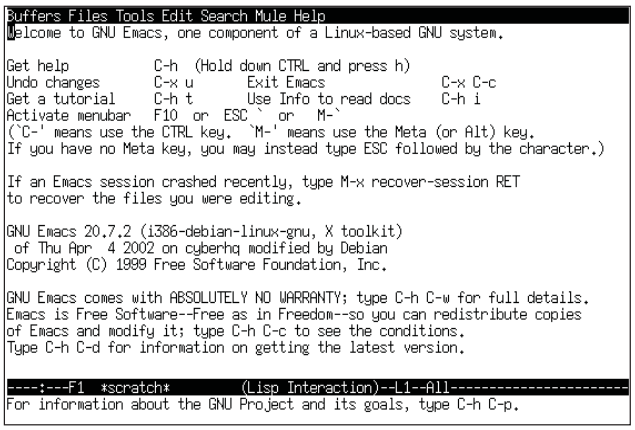
Az almódok, azaz a bővítmények a főmódokat egészítik ki. A legtöbb almód a főmódtól függetlenül működik, így külön-



4. kép Két fájl összehasonlítása. Az Emacs nemcsak a megváltozott sorokat mutatja, hanem a sorokon belüli különbségeket is



5. kép A szerző .emacs állományának szerkesztése az Emacsben, példa az Emacs Lispre



6. kép Az Emacs karakteres felületen indítva

féle dokumentumokon használható. Például a Show Paren-mód a zárójeleket párosítja össze. Ez a C-programozók számára is hasznos, még Text-módban is, de Lisp-programozók számára nélkülözhetetlen segédeszköz.
Az almódokat tetszés szerint ki- és bekapcsolhatjuk. Programozásnál például az Auto Fill-mód (szövegkitöltés, sortörés, bekezdések) hasznos a megjegyzések írásánál, de idegesítő a kód írásánál.
Bizonyos almódok mindenre kiterjedők, vagyis az összes szövegtárolóra vonatkoznak, mások csak az adott szövegtárolón belül érvényesek. Egy adott mód bekapcsolásához a neve után a `-mode` karakterláncot írjuk a parancssorba. Például a Parentheses-mód indításához nyomjuk le a M-X billentyűkombinációt, majd írjuk be a `show-paren-mode` parancsot. A mód kikapcsolásához újra ezt a parancsot futtassuk.
Néhány hasznos szövegtárolón belüli almód: Abbrev-mód (menet közbeni önműködő javítás), Auto Save-mód, Font-Lock-mód (színes szövegkiemelés), Flyspell mód (menet közbeni helyesírás-ellenőrzés) és Overwrite-mód. Két hasznos almód, ami minden szövegtárolóra érvényes, a Line Number és a Column Number mód. Ezek a kurzor pillanatnyi helyzetét írják ki a módsorba, általában a jobb oldalra.
Egy további hasznos mód az Ispell, amivel a szövegtároló helyesírás-ellenőrzését végezhetjük el. Külön almódokkal rendelkezik elektronikus levelek, programszövegekbe ágyazott megjegyzések és karakterláncok, illetve más különleges alkalmazási területek ellenőrzésére.

A .emacs fájl

Az Emacs testreszabásának kulcsa a *~/.emacs* beállítóállomány. A rendszergazdák általában egy egész rendszerre érvényes beállítóállományt készítenek. Ha ez nekünk nem felel meg, a saját beállítóállományunkban megkérhetjük az Emacset, hogy ne vegye figyelembe. Hibakereséskor hasznos lehet, ha az `emacs -q` parancsot kiadva az Emacsot a beállítóállomány elolvasása nélkül indítjuk el. A beállítóállomány nem más, mint egy kis Elisp-program, ami tetszésünk (vagy a rendszergazda tetszése) szerint állítja be az Emacset (*5. kép*).
A beállítóállományban változóknak is adhatunk értéket. Én az alábbi módon állítottam be a HTML Helper-mód egyes változóit:

```
(setq html-helper-do-write-file-hooks t)
(setq html-helper-build-new-buffer t)
(setq html-helper-address-string "<a
href=\"mailto:ccurley@trib.com\">Charles
Curley</a>")
```

Rövid, hasznos függvények vagy makrók is bekerülhetnek a *.emacs* állományba, például a következő makró a mai dátumot szúrja be:

```
(defun insert-date ()
  "A mai dátum beszúrása az \"insert-date-
   ➥format\" változó szerint."
  (interactive "*")
  (insert (format-time-string insert-date-
          ➥format
                              (current-time))))
```

A makrókhoz gyorsbillentyűket vagy billentyűkombinációkat is rendelhetünk. Ezek után a billentyűkombinációval is elindítható a makró; például a dátumbeszúró makró megírása után az F3 billentyűt a következő sorral rendelhetjük hozzá:

```
(global-set-key [f3] ´insert-date)
```

Ez a módszer arra is alkalmas, hogy megváltoztassuk a billentyűzetkiosztást. Ha nem tetszenek az Emacs bizonyos hosszú billentyűkombinációi, megváltoztathatjuk őket.
Az Emacs testreszabásának másik módja a ***Customize*** menü használata, amit a `M-X customize` parancs vagy az ***Option***s legördülő menü segítségével hívhatunk elő. A kiterjedt menürendszerben a felhasználók megváltoztathatják a változók értékét, és a változásokat menthetik a saját beállítóállományukba.

Az Emacs mint kiszolgáló

Számos program, például a `mutt` és a `crontab` külső programot hív meg szövegszerkesztőként. Az Emacs használatát úgy engedélyezhetjük, hogy az Emacset kiszolgálóként indítjuk. Írjuk be a következő sort a ***.emacs*** állományba:

```
(server-start)
```

Ezt követően állítsuk be az `EDITOR` vagy a `VISUAL` környezeti változót az `emacsclient` értékre. Bash használata esetén a következő sort adjuk hozzá a */etc/bashrc* állományhoz vagy a saját ***~/.bash_profile*** állományunkhoz:

```
export VISUAL=emacsclient
```

Ezek után a `crontab -e` végrehajtásakor vagy üzenet szerkesztésekor a `mut`-ban a meglévő Emacs-munkamenetben szerkeszthetünk, ahelyett, hogy egy új Emacs-példány indulására kellene várnunk. A szerkesztés befejezésekor a CTRL-C K helyett a CTRL-C # paranccsal fejezhetjük be a szövegtároló szerkesztését, és léphetünk ki az `emacsclient` programból. Az `emacsclient` működéséhez az Emacs programnak – amikor a külső program meghívja – már futnia kell. Ez összhangban van az Emacs használatának javasolt módjával, ami szerint bejelentkezéskor indítsuk el az Emacs egy példányát, és amíg ki nem jelentkezünk, hagyjuk futni. Az `emacsclient` használatának egyik eredménye az, hogy egyszerre csak egy Emacs-példány fut. Bár a memória napjainkban olcsó, ez nem volt mindig így. Még ma is, ha a Linuxot hordozható számítógépen vagy régi számítógépen futtatjuk, a takarékos memóriafelhasználás jó dolog.

Érdemes beállítani, hogy az Emacs Mail-módban szerkessze a leveleket. Ha `mut`-tot használunk, a következő sorokat adjuk hozzá a ***.emacs*** állományhoz:

```
;; Ha a fájlnév /tmp/mutt kezdetű, lépjünk
;; mail-mode üzemmódba
(setq auto-mode-alist (append (list (cons
➥"^\/tmp\/mutt" ´mail-mode))
                              auto-mode-alist))
```

Természetesen szeretnénk megfelelni az RFC-kben leírt netikettnek is, ezért engedélyezzük az Auto Fill-módot a levél szerkesztésekor. A legtöbb főmód olyan, hogy a módba történő belépéskor és módból való kilépéskor végrehajthatunk egy parancsot. A következőképpen utasíthatjuk a Mail-módot arra, hogy belépjen az Auto Fill-módba:

```
(defun my-mail-mode-hook ()
  (auto-fill-mode 1)
  )
(add-hook ´mail-mode-hook
          ´my-mail-mode-hook)
```

A levél megírása után, ha idegesíteni szeretnénk az NSA-t, használjuk a Spookot. Ha tiltakozni szeretnénk a Communications Decency Act ellen (ez helyes cselekedet), és jó pár amerikai politikust szeretnénk idegesíteni, akkor nézzük meg a Bruce-t.

Végül mielőtt befejeznénk ismerkedésünket e vad és buja szövegszerkesztővel, hadd hívjam fel a figyelmet a */etc* könyvtárra (az Emacs könyvtárfában). Ez a könyvtár sok hasznos dokumentumot tartalmaz, például az Emacs angol nyelvű referenciakártyáját forrásban (***refcard.tex***) és PostScriptben (***refcard.ps***). A referenciakártya különböző nyelvű fordításai is megtalálhatók itt. Van még itt némi háttéranyag az Emacsről és a GPL egy példánya.

Van egy dolog, amit ritkán találunk meg (legfeljebb véletlenül) a szabadalmaztatott programokban, de az Emacsben jelen van: a humor. Nézzük meg a hibajelentést a 2199-es évből, a Spook-mód szólistáját, az Emacs rövidítés néhány magyarázatát stb. Végül, ha igazán szeretnénk megdolgoztatni a betűkészlet-kiszolgálónkat, nyissuk meg a HELLO állományt.

Linux Journal 2003. március, 107. szám

Charles Curley
(w3.trib.com/~ccurley) Linux-oktató és szakíró.
Természetesen ezt a cikket is Emacsszel írta, Linuxon.

KAPCSOLÓDÓ CÍMEK

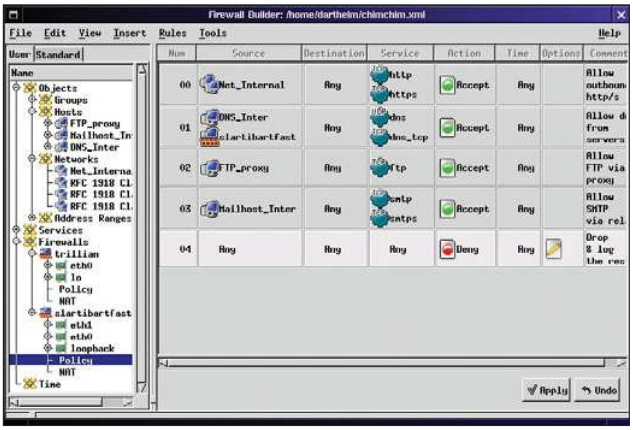
Az Emacs beépített súgórendszere CTRL-H
A */etc* könyvtár az Emacs-fában
A GNU Emacs honlapja
➲ http://www.fsf.org/software/emacs/emacs.html
A szerző *.emacs* állománya
➲ http://w3.trib.com/~ccurley/emacs.init.html
Az Emacs-webgyűrű
➲ http://www.gnusoftware.com/WebxRing
Az „emacs for vi Users” ➲ http://grok2.tripod.com
A „GNU Emacs Tutorial” *Sarir Khamsi*-tól (régi, de hasznos)
➲ http://www.futureone.com/~sponge/tutorial/emacs/index.html
A „very unofficial dotemacs home” *Ingo Koch*-tól
➲ http://www.dotemacs.de.
A „The Emacs Beginner's HOWTO” *Jeremy D. Zawodny*-tól ➲ http://www.tldp.org/HOWTO/Emacs-Beginner-HOWTO.html
A „How do I make common modifications to my Gnu Emacs .emacs file? GNU Emacs Manual – Init Examples”, rupe ➲ http://www.yak.net/fqa/124.html
Emacs-tippek: „Nifty ways to get your work done in /the/editor” ➲ http://www.portico.org/index.php3?catList=28

A cikkben ismertetett módok, amelyek nem részei az Emacsnek
HTML Helper-mód
➲ http://www.gest.unipd.it/~saint/hth.html
Java Development Environment for Emacs (JDEE)
➲ http://jdee.sunsite.dk
PSGML-mód
➲ http://www.lysator.liu.se/projects/about_psgml.html
LDP Author Guide, Mark F. Komarinski, Jorge Godoy és David C. Merrill munkája ➲ http://www.tldp.org/LDP/LDP-Author-Guide. Az SGML és az XML telepítésében és használatában segít, különösen a DocBookében, továbbá az Emacs testreszabását is ismerteti.
Emacs/W3 v4.0 ➲ http://www.cs.indiana.edu/elisp/w3

A Firewall Builder használata (1. rész)

Gondoltál már arra, hogy egyetlen könnyen használható, grafikus alkalmazással az összes tűzfalad és kiszolgálód házirendjét kézben tarthatnád?

A 2.4-es Linux Netfilter tűzfalkódja, valamint felhasználói felülete, az IP Tables alaposan kiérdemelte népszerűségét és az őt érő dicséreteket. Ezek segítségével válhattak a Linux alapú tűzfalak a kereskedelmi, állapotalapú csomagszűrő tűzfalakkal egyenértékű megoldásokká, akár a szolgáltatások vagy az intelligencia, akár a biztonság szempontjából vizsgáljuk őket.
A Netfilter kapcsán egyetlen hiányosságot lehetett megemlíteni: a felhasználóbarát jelleg elmaradását. Ha egy tűzfalat jó grafikus felülettel sikerül ellátni, az nem csak a kisebb műszaki érzékkel bíró felhasználók számára jelent segítséget. Idővel még a leginkább kockafejűek is rájönnek, hogy gyorsabban és kevesebb hibával készíthetnek tűzfalházirendeket, ha szemléletes megjelenítés és jelzések segítik őket munkájukban. Az IP Tables-szabályok írásmódjában egyértelműen a biztonsági szabályok szempontjai érvényesülnek, nem az olvashatóságé.
A Firewall Builder (*1. kép*) igazán kiváló grafikus tűzfalfelület. Segítségével állomás-, hálózat- és szolgáltatásobjektumokat hozhatunk létre, amelyeket tetszőleges számú tűzfalszabályban (újra)felhasználhatunk. A szabályokat szemléletes és áttekinthető módon jeleníti meg, és mivel alapvetően operációs rendszertől független, a Firewall Builder segítségével nemcsak Netfilter/IP Tableshez, de a FreeBSD IP Filteréhez, az OpenBSD `pf`-éhez és a Cisco PIX-tűzfalakhoz is készíthetők szabályok.
Ez alkalommal, illetve a következő részben azt mondom el, hogyan szerezheted be és telepítheted a Firewall Buildert, majd szót ejtek arról is, hogy a segítségével miként hozhatsz létre szemléletes és egyszerű módon IP Tables-szabályokat. Elsőként a Firewall Builder telepítését tárgyalom, ezt követően feltöltjük az objektumokat tároló adatbázisát. A szabályok létrehozásáról a következő részben fogok írni.



1. kép A Firewall Builder működés közben

Hová kerüljön a Firewall Builder?

Elsőként tekintsük át, hova kell telepíteni, illetve honnan kell futtatni a Firewall Buildert. Én úgy gondolom, hogy a Firewall Buildert nem célszerű magán a tűzfalgépen, illetve bármely egyéb nyilvánosan elérhető, megerősített gépen, azaz úgynevezett bástyagépen futtatni. Nyilván senki nem gondolt rá, hogy X Window fusson egy ilyen gépen.
A Firewall Buildert egy mindennapos használatra szolgáló munkaállomáson érdemes használni. Az így létrehozott tűzfalparancsfájlokat utolsó lépésként `scp`-n vagy más biztonságos eljárással át kell másolni az őket alkalmazó gépre. A Firewall Buildert eleve ilyen használatra tervezték.
Másrészről viszont, ha a Firewall Buildert egy adott állomás – például egy Linux 2.4-es alapú webkiszolgáló – helyi védelmét szolgáló Netfilter-parancsfájlok létrehozására akarjuk használni, akkor talán nem túl nagy vétség, ha magát a Firewall Buildert is a parancsfájlokat alkalmazó gépen futtatjuk. Az X11 telepítésére ekkor is ügyelni kell, illetve az adott állomásnak megfelelően beállított tűzfal mögött kell lennie.
Fontos, hogy a Firewall Buildert nem szükséges az összes beállítani kívánt állomáson külön futtatni. Nem muszáj tehát egyetlen olyan gépen sem futtatni, amelyen egyébként nem használnál X Window rendszert. Egyetlen Firewall Buildert futtató gépen számos más gép számára hozhatsz létre tetszés szerinti szabályokat. Ennek pontos módjáról rövidesen szót ejtünk.

A Firewall Builder beszerzése és telepítése

A Firewall Builder Projectnek természetesen saját honlapja van, ahonnan a legújabb kiadás, illetve a leírás letölthető (➲ http://www.fwbuilder.org). Ha a weblapon és az itt olvasottak között bármilyen eltérés tapasztalható, akkor az előbbi a mérvadó. A Firewall Buildernek a honlapon található telepítési leírása érthető és pontos. Természetesen semmilyen változás nem zárható ki cikkem írása és a megjelenés időpontja között.

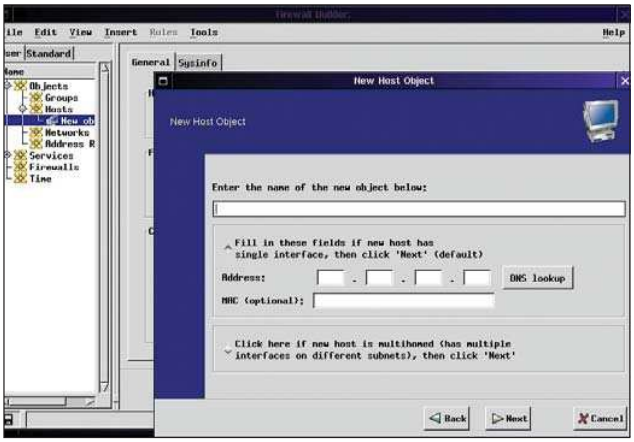
Debian

A könnyebb esettel kezdem. Debian 3.0 alatt a Firewall Builder közvetlenül a Debian telepítési forrásból tehető fel, a Debian ugyanis `fwbuilder` név alatt saját, hivatalos támogatású debcsomaggal rendelkezik. Ez a csomag egyebek mellett a következő Debian-csomagoktól függ: ***libfwbuilder0***, ***fwbuilder-iptables***, ***libgtk1.2***, ***libgtkmm1.2***, ***libxslt1***, ***libxml2*** és ***libsnmp4.2***. A teljes függőségi listát – ha nem gond – most elhagynám.
Ha az `fwbuilder` telepítése `apt-get`-tel történik, akkor az `apt-get` minden szükséges csomag azonosításáról és telepítéséről gondoskodik. A Debian `fwbuilder-doc` csomagjának telepítését is javaslom, és bár a felrakása nem kötelező (és ilyen módon nem is történik meg önműködően, hiszen általa az `apt-get` nem tud semmilyen függőséget feloldani), mindenre kiterjedő és hasznos leírást találunk benne.

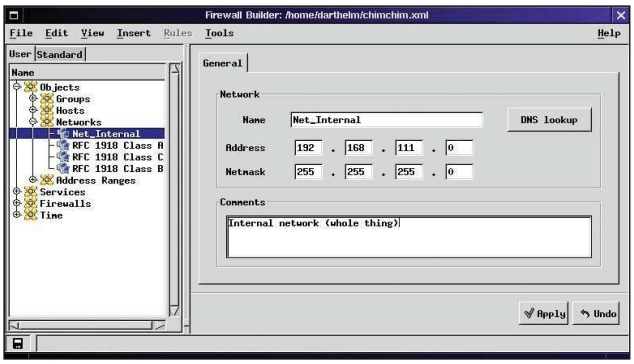
Red Hat

A Red Hat 8.0 (cikkem írásakor ez a legújabb Red Hat-változat) terjesztésnek hivatalosan még nem része a Firewall Builder.
A Firewall Builder fejlesztői csapat ugyanakkor számos Red Hat-terjesztéshez készített RPM fájlokat – ezeket a Firewall Builder

➲ http://sourceforge.net/project/showfiles.php?group_id=5314 letöltések oldalán találhatjuk meg
Az *fwbuilder* és a *libfwbuilder* csomagra lesz szükségünk, illetve az *fwbuilder-ipt*, *fwbuilder-ipf* vagy *fwbuilder-pf* csomag valamelyikére, attól függően, hogy Linux Netfilter/IP Tables, FreeBSD IP Filter vagy OpenBSD `pf` számára akarunk-e szabályokat létrehozni. Az előbbi három csomagból akár mindegyiket is feltelepíthetjük, ha szükséges. Mivel a Firewall Builder végeredményként ASCII formátumú parancsfájlokat állít



2. kép Az Insert Host (állomás beszúrása) párbeszédpanel



3. kép Az Insert Network (hálózat beszúrása) párbeszédpanel

elő, Linux alatt más operációs rendszerek számára is nyugodtan létrehozhatunk szabályokat.
A Firewall Builder-csomagok telepítése előtt a következő normál Red Hat-csomagoknak kell jelen lenniük a rendszerben: bind-utils, gdk-pixbuf, glib, glibc, gtk+, gtkmm, libfwbuilder, libsigc++, libstdc++, libxml2, libxslt, openssl-0.9.6b, ucd-snmp és XFree86-libs.
Ezek mellett szükség lesz a gtkmm (the GIMP Tool Kit Minus Minus) csomagra is, ami a GTK+ C++-kötéseit tartalmazza. A csomag a Ximian Gnome része, azonban a ➲ http://www.freshrpms.net címről is letölthető.

SuSE

A Red Hathez hasonlóan jelenleg még a SuSE sem építette be a Firewall Buildert a hivatalos terjesztésbe. A SuSE 8.1 RPM-ek (nem hivatalos, külső forrásból származók) a Firewall Builder letöltések oldalról (➲ http://sourceforge.net/project/showfiles.php?group_id=5314) érhetők el.
Az *fwbuilder* és a *libfwbuilder* csomagra mindenképpen szükség lesz, illetve az *fwbuilder-ipt*, az *fwbuilder-ipf* és az *fwbuilder-pf* csomagok közül egyet vagy többet úgyszintén le kell tölteni. A telepítéshez a következő normál SuSE-csomagok szükségesek: gcc, gdk_pixbuf, glib, glibc-2.2.4, gtk, gtkmm, libsigc++, libstdc++, libxml2, libxslt, libz, openssl-0.9.6b, ucdsnmp és xshared.

Objektumok létrehozása

A csomagok telepítése után a Firewall Builder készen áll a használatra. Mindössze egyetlen parancsot kell megjegyezni: a `fwbuilder`-t. A parancs kiadásakor az X Window rendszernek már futnia kell. A program nem csak rendszergazdaként használható, sőt nem is javaslom, hogy különösebb indok nélkül így futtassuk, hiszen ki tudja, mit nézünk el.
Az *fwbuilder* ablak megnyílása után nekiláthatunk az objektumok létrehozásának (*2. kép*). A Firewall Builder felfogásának az az alapja, hogy a szabályokat újrafelhasználható, húzd és ejtsd módszerrel elhelyezhető objektumok segítségével hozzuk létre, tehát az objektumoknak még a szabályok megalkotása előtt rendelkezésre kell állniuk. Még a Firewall Builder önműködő szabálylétrehozó varázslói sem használhatók, ha nem hoztuk létre a szükséges objektumokat.
Az objektumok hálózati állomásokat, hálózatokat (ezeket IP-cím és alhálózati maszk azonosíthatja), címtartományokat, TCP/IP-szolgáltatásokat, tűzfalakat (többlaki tűzfalrendszereket és bástyagépeket), időtartományokat és más objektumok csoportjait képviselhetik. Mindenki tetszése szerinti mennyiségben hozhat létre objektumot – annyit, amennyire szüksége lesz a saját szabályaiban. Értelemszerűen legalább egy tűzfal- és legalább egy hálózat- vagy állomásobjektumra szükség van. Számos általánosan használt TCP/IP-szolgáltatáshoz előre megadott objektumokat találunk.

Hálózati állomásobjektumok

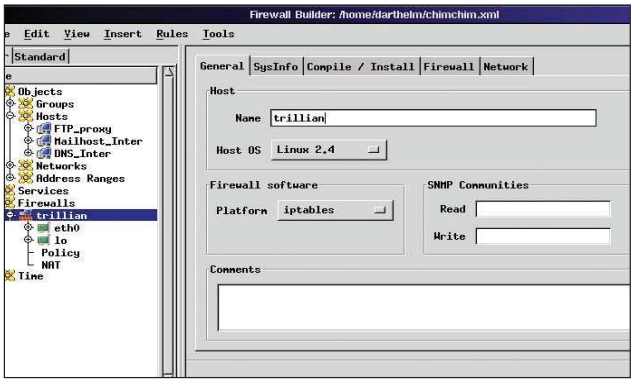
Az objektumok a Firewall Builder ***Insert*** (beszúrás) menüjével hozhatók létre. A 2. képen az állomások létrehozására szolgáló ***Insert host*** (állomás beszúrása) párbeszédpanel látható. A szabályok létrehozásakor az állomások legfontosabb jellemzője az IP-címük. Ha az állomásokat a MAC-, illetve az ethernetcímük alapján azonosító szabályokat akarunk írni, akkor ilyen címeket is megadhatunk. Mint a képről is kitűnik, az IP-címek kézzel és DNS-keresés alapján is megadhatók. Az utóbbi szolgáltatás ugyan hasznos, de ne feledjük, hogy csak olyan állomások esetében használható, amelyek címe a Firewall Buildert futtató gépről feloldható.

Hálózatobjektumok

A *3. képen* az ***Insert network*** (hálózat beszúrása) párbeszédpanel látható. Az állomás beszúrásával ellentétben – ami külön ablakban jelenik meg – a hálózat beszúrását egy egyszerű ***New object*** (új objektum) űrlapon végezhetjük el, a főablak jobb oldali részén. Az űrlap tulajdonképpen egyszerűbb, mint az állomás beszúrására szolgáló párbeszédpanel, mindössze a megfelelő hálózat IP-címét és alhálózati maszkját kell megadnunk, nevet kell adnunk az objektumnak, és szükség szerint megjegyzéssel is elláthatjuk.

Tűzfalobjektumok

Az objektumok közül messze a legbonyolultabb a tűzfalobjektum. Magukon az alapbeállításokon könnyedén túl lehet jutni, mindössze a tűzfal hálózati felületét vagy felületeit kell megadni az IP-cím és az alhálózati maszk által. A tűzfalobjektum hozzáadása és az *fwbuilder* ablakának bal oldali, a felhasználói objektumokat tartalmazó listában való megjelenése



4. kép A tűzfal tulajdonságai

után kattintsunk rá az ikonjára. Öt adatlap (fül) fog megjelenni az ablak jobb oldalán (*4. kép*).

A ***General*** (általános) lapon a tűzfalobjektum létrehozásakor megadott állomásnevet láthatjuk. Fontos, hogy a megfelelő ***Host OS*** (futtató operációs rendszer) és Platform (***célrendszer***) beállításokat megadjuk, hiszen a Firewall Builder így tudja majd kiválasztani a megfelelő fordítómotort, amikor a szabályokat az adott tűzfal számára lefordítja.

A SysInfo (***rendszerinformációk***) csak az SNMP-vel kapcsolatos beállításokat tartalmaz (lásd még a széljegyzetet). A ***Compile/Install*** (fordítás/telepítés) lap a tűzfalházirend önműködő telepítésének beállításait tartalmazza. Ha a telepítést kézzel akarjuk végezni, a lap tartalmával nem kell foglalkoznunk. Valamikor a – remélhetően nem túl távoli – jövőben a Firewall Builder képes lesz arra is, hogy önműködően, SSL felett továbbítsa és telepítse a tűzfalparancsfájlokat. Írásom elkészültekor az `fwbd` démon, amit ennek a szolgáltatásnak a használatához majd a céltűzfalon kell futtatni, még nem jelent meg.

A ***Compile/Install*** lap ***Installer*** (telepítő) beállítását az alapértelmezett `fwbd` értéken is hagyhatjuk, ekkor – a szolgáltatás támogatásának hiánya ellenére – sem fog baj történni, a lefordított tűzfalszabályokat a program a kezdőkönyvtárunkba menti. A ***Rules*** (szabályok) menü ***Install*** (Telepítés) eleme természetesen szürkén jelenik meg. Ha viszont úgy döntünk, hogy az ***Installer*** beállításnak ***Install Script*** (parancsfájl telepítése) értéket adunk, akkor a ***Policy Install Script*** (szabálytelepítő parancsfájl) mezőben saját parancsfájlunk elérési útját írhatjuk be, illetve parancssori átadott értékeket is megadhatunk hozzá. A saját parancsfájl futtatására akkor kerül sor, amikor a szabályok lefordítása után a ***Rules/Install*** (szabályok/telepítés) parancsot választjuk.

Ezzel a módszerrel kényelmesen, parancsfájlból indíthatunk például `scp`-t, ami elvégzi a szabályok másolását a céltűzfalra. Ilyen telepítő parancsfájlra példákat is találhatunk a Firewall Builder letöltések oldalán (➲ http://sourceforge.net/project/showfiles.php?group_id=5314); közülük is az `fwb_install` érdemes kiemelt figyelemre.

A telepítőbeállítástól függetlenül a Firewall Builder a lefordított parancsfájlokat egy helyi ASCII-fájlba írja, aminek egy, a tűzfalobjektuméval megegyező nevet ad, kiterjesztésnek pedig a *.fw*-t választja. Például a 4. képen is látható Trillian nevű tűzfalhoz készített parancsfájlokat ***trillian.fw*** név alatt menti.

Folytatva a tűzfalobjektum tulajdonságainak vizsgálatát: a ***Firewall*** (Tűzfal) lap szolgál a ***General lapon*** kiválasztott célrendszerre – ami ez esetben a Netfilter/IP Tables – egyedileg jellemző beállítások megadására (*5. kép*). Az alapértelmezett beállítások valószínűleg a legtöbb felhasználónak megfelelnek, néhány lehetőséget mégis érdemes áttekinteni.

A tűzfalak és az SNMP

A Firewall Builder kiterjedt Simple Network Management Protocol (SNMP) támogatással rendelkezik. Az SNMP könnyen használható eszköz az SNMP-képességekkel rendelkező hálózati eszközök és állomások beállításainak lekérdezésére, illetve beállításaik frissítésére, feltöltésére (a Firewall Builder egyébként csak lekérdezést végez). Én biztonságra törekvő környezetben soha sem szerettem az SNMP-t használni. Az SNMP-átvitelek hitelesítése közösségi karakterláncok, avagy jelszók segítségével történik, amiket a felek mindenféle titkosítás nélkül, nyílt szövegként továbbítanak. Emiatt egy általános, megosztott átviteli közeget használó – például kapcsoló nélküli ethernet vagy kábelmodemes – hálózatban nem túl bonyolult feladat az SNMP-jelszavak lehallgatása, sőt egyes esetekben ez még kapcsolóval rendelkező ethernethálózaton is megoldható. Az SNMP tehát nem megbízható hálózatokban meglehetősen kockázatos eszköz a berendezések beállításainak módosítására, és még részben megbízható hálózatokon sem feltétlenül jó választás. Ne feledjük, a hálózat biztonságát a legtöbb esetben belső személyek fenyegetik.

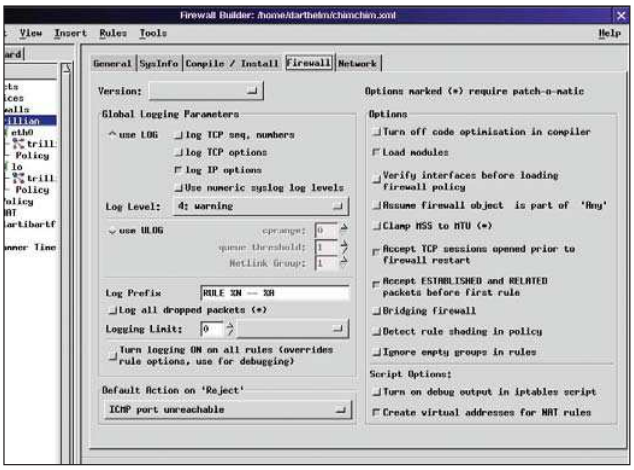
További adalék, hogy a legtöbb Linux-terjesztésben megtalálható UC-Davis SNMP-csomagban korábban jó néhány biztonsági hiányosságot találtak. Bástyagépen, tűzfalon tehát gyakorlatilag semmilyen körülmények között nem javaslom, hogy ezt az SNMP-démont – vagy bármilyen más fajtát – futtassuk. Az, hogy a Firewall Buildernek szüksége van az SNMP-könyvtárakra, nem okoz különösebb gondot, hiszen – mint már írásom elején említettem – a Firewall Buildert nem magán a tűzfalon vagy a bástyagépen kell futtatni.

Természetesen mindenki maga dönti el, hogyan és mekkora mértékben használja ki a Firewall Builder SNMP-szolgáltatásait. A hogyannal azonban – álláspontommal összhangban – én nem fogok foglalkozni.

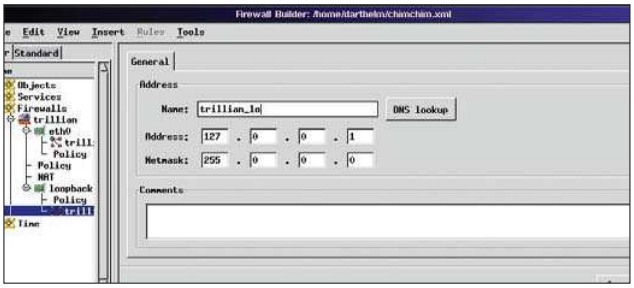
A ***Global Logging Parameters*** (általános naplózási beállítások) lap segítségével szabályozhatjuk, hogy a Firewall Builder hogyan készíti a naplóbejegyzéseket. Az alapértelmezett ***Log Level*** (naplózási szint) a ***6***-os (Info is okay). Jómagam csak az eldobott és visszautasított csomagokat naplózom, vagyis a 4-es (Warning) szintet használom.

A ***Firewall*** (Tűzfal) ablakban érdemes engedélyezni az ***Assume firewall object is part of Any*** (Feltételezze, hogy a tűzfalobjektum része a Bármely csoportnak) beállítást. A beépített ***Any*** (bármely) ***Source/Destination*** (forrás/cél) objektum esetében az alapértelmezett értelmezési mód a „bármely hálózati állomás, kivéve a tűzfalat”. A tűzfalak szabályainak létrehozásakor ez nem szokatlan elgondolás, de időnként meglepő viselkedést válthat ki. Ha például a házirend utolsó szabálya egy source=any, destination=any, service=any, action=drop és logging=on szabály, akkor nyilván azt várjuk, hogy a rendszer elhárít és naplóz minden, a tűzfalhoz való, az előbbi szabályoknak meg nem felelő kapcsolódási kísérletet. A rendszer valóban eldobja őket, de nem emiatt a szabály miatt. Elvetésükre az `INPUT` lánc alapértelmezett házirendje miatt kerül sor, amit a Firewall Builder mindig `DROP` értékre állít. Az említett szabály tehát csak a tűzfalon keresztüli kapcsolódási kísérletek esetében jut szerephez. Mivel magát a tűzfalat nem tekintjük az ***Any*** csoport részének, a fenti szabály csak a `FORWARD` láncban lép életbe, az `INPUT` vagy az `OUTPUT` láncban nem.

Az ***Assume firewall object as part of Any*** beállítás megfordítja ezt, és a fenti szabály végre úgy fog működni, ahogy azt erede-



5. kép Célrendszertől függő tűzfaltulajdonságok



6. kép A hurokfelület címének megadása

tileg szerettük és vártuk volna. Ugyanakkor figyelembe kell venni, hogy más dolgokat viszont bonyolultabbá tesz, például a tűzfal hálózati felületeire vonatkozó címhamisítás-védelmi szabályokat. Tehát: az éremnek két oldala van. Én ezt a beállítást inkább kikapcsolva hagyom. Ezután úgy alakítom a Firewall Builder-parancsfájlokat, hogy legalább az `INPUT` lánc esetében tartalmazzák a `log` és `drop` sorokat, vagy hozzáadok egy külön tűzfalbemeneti `log` és `drop` szabályt a házirendhez. Ha valami nem biztos, próbálgatni kell, majd szükség szerint javítani. Ilyenkor jó szolgálatot tehet a ***Global Logging Parameters*** rész ***Log all dropped packets*** (Naplózzon minden eldobott csomagot) beállítása, bár a működéséhez a Netfiltert a ***Patch-O-Matic Dropped Table*** folttal kell lefordítani – ha a Linux-terjesztésünk alapváltozat szerinti rendszermagját használjuk, akkor ez valószínűleg nincs benne.

A tűzfalobjektum tulajdonságait tartalmazó lapok közül a ***Network*** (***hálózat***) az utolsó. Ez a ***General*** lapon kiválasztott futtató operációs rendszerrel kapcsolatosan tartalmaz beállításokat. Az itt található beállítások a rendszermag viselkedését közvetlenül befolyásolják – ha ez valakit megrémít, nyugodtan átugorhatja ezt a lapot. Egy apróság: ha jelentős terheléssel küzdő, egy teljes hálózat védelmét biztosítani hivatott tűzfalról, és nem csupán egy megerősített, de egyszerű gépről van szó, akkor nem árt bekapcsolni a ***Packet Forwarding*** (csomagtovábbítás) beállítást.

Hurokfelületek

Bármilyen hihetetlen, de még mindig nem végeztünk a tűzfalobjektum beállításával. A 4. képet szemlélve talán feltűnt, hogy az ablaknak az objektumokat hierarchikusan megjelenítő bal oldali részében a Trillian nevű tűzfal két felülettel bír, ez az `eth0` és a `lo`. Az `eth0` hálózati csatoló alág a tűzfalobjektum beszúrásakor önműködően létrejött, ellenben a `lo` felületet – ami a Trillian hurokeszközét jeleníti meg – kézzel kellett létrehozni. Kicsit furcsa, hogy a létrehozásáról nem gondoskodik magától a program. Minden tűzfal, még a többlaki rendszerek vagy a bástyagépek esetében is szükség van olyan szabályokra, amelyek lehetővé teszik a hurokeszközök működését, és ezzel megelőzik a helyi folyamatok megszakítását.

Ha hurokfelületet akarunk létrehozni, válasszuk ki a listából a tűzfalobjektum ikonját, nyissuk meg az ***Insert*** menüt, majd válasszuk az ***Interface*** (felület) parancsot. Az ***Interface*** pont mindaddig szürke marad, amíg egy állomás- vagy tűzfalobjektumot ki nem választunk. A tűzfalobjektum alatt egy új felületikon jelenik meg, az új felület tulajdonságai pedig a jobb oldalon jelennek meg. Írjuk be a felület nevét a ***Name*** (név) mezőbe (példa: lo), majd kapcsoljuk ki a ***This Interface is External*** (külső felület) beállítást, ez ugyanis csökkentené a biztonságot. A beállítást csak külső felületek és DMZ-felületek objektumain kell engedélyezni.

Következő lépésként, miközben az új felület objektuma ki van választva, újra nyissuk meg az ***Insert*** menüt, és válasszuk az `Address` (cím) parancsot. Egy cím alobjektum bukkan fel az új felület alatt, jobb oldalon pedig a tulajdonságai jelennek meg (*6. kép*). Adjunk meg egy nevet, IP-címként 127.0.0.1-et, hálózati maszkként pedig 255.0.0.0-t (az utóbbit a program magától is beírja). Bizonyos helyzetekben a rendszer több hurokfelülettel is rendelkezhet, ilyenkor a megadott cím más is lehet (127.0.0.2 stb.). Az esetek túlnyomó részében csak egy ilyen felület van, és ennek IP-címe 127.0.0.1. Ha nem vagyunk biztosak a dolgunkban, a tűzfalgépen adjuk ki az `ifconfig` -a parancsot.

Ha az összes objektum megadásával végeztünk, vagy legalábbis eleget gyűjtöttünk össze ahhoz, hogy a szabályokat elkészíthessük, a ***File*** (fájl) menü `Save` (mentés) parancsával mentsük az objektumokat. Az alkalmazás egy fájlnevet kér, a fájlt pedig ***.xml*** kiterjesztéssel a kezdőkönyvtárunkba fogja menteni. Egyes parancsfájlok azt várják, hogy az objektumokat ***objects.xml*** névvel mentsük és a ~ könyvtárban tároljuk, de ezt módosítani lehet. Más szavakkal: úgy nevezzük el az objektumokat tartalmazó fájlt, ahogy tetszik, és oda mentjük, ahova akarjuk. A nevét és a helyét azonban ne felejtsük el, hiszen ha az `fwb_installt` akarjuk módosítani, vagy más házirend-telepítő parancsfájlt akarunk készíteni, még szükségünk lesz ezekre az adatokra.

A következő lépések a következő hónapban

Mindenki szüksége szerint, az adott környezetnek megfelelően hozzon létre további állomásokat, hálózatokat és tűzfalobjektumokat. A ***Network Range*** (hálózattartomány) és a ***Time*** (idő) objektumokról nem szóltam, ám mindkettő használatát könnyű megérteni – ha máshogy nem, kísérletezni kell velük egy kicsit, vagy bele kell olvasni a ➲ http://www.fwbuilder.org címen található leírásba. A jövő hónapban szabályokat fogunk létrehozni a már meglévő objektumok felhasználásával. Addig is már megszerzett tudásunk alapján próbáljunk tovább ismerkedni a Firewall Builderrel.

Jó szórakozást!

Linux Journal 2003. május, 109. 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”.

Héjprogramozás Linux alatt – feltételek (3. rész)

Cikksorozatunk előző két részében egy apró programot fejlesztettünk lépésről lépésre.

Célunk az volt, hogy megszámoljuk, hány alkönyvtár nyílik egy adott könyvtárból. Óriásinak egyelőre nem nevezhető fejlesztésünk jelenleg itt tart:

```
1: #!/bin/sh
2: lista=`ls -l $1 | grep ^d`
3: echo "$lista"
4: echo "Összesen"`echo -n "$lista" | wc -l`"
        ➥alkönyvtár"
```

A második sorban a `lista` nevű változóban összegyűjtjük az első parancssori értékben megadott könyvtár alkönyvtárainak nevét (és számos egyéb adatát). A harmadikban ezt a listát kiíratjuk, a negyedikben pedig megszámoljuk, hány sorból áll. Ez utóbbiban azért használtuk az `echo` parancs `-n` kapcsolóját, mert a program akkor is egy alkönyvtárat számolt meg, amikor valójában egy sem volt. Rájöttünk, hogy ezt az `echo`-nak az a furcsa szokása okozza, hogy akkor is kiküld egy újsorkaraktert, ha valójában nincs is mit kiírnia. A `wc` viszont éppen az újsorkarakterek alapján tájékozódik…

Újabb gondok

Egy program fejlesztése során gyakran támad olyan érzésünk, hogy most aztán már semmi baj nem érhet minket, hiszen mindenre gondoltunk. Aztán rendszerint mégis kiderül valami. Esetünkben is ez a helyzet, sőt rögtön két gondunk is akad. Először is még semmit nem tettünk annak érdekében, hogy kiszűrjük a nem létező könyvtárakat. Márpedig a felhasználónak bármilyen badarságot „jogában áll” parancssori értékként megadni. A második gond ennél összetettebb. Az előző részben ugyan nagy megelégedésünkre szolgált az `echo -n` kapcsolójának felfedezése, hiszen kijavítottunk vele egy hibát. Csakhogy ha közelebbről megvizsgáljuk programunk működését, hamar rájövünk, hogy most meg mindig eggyel kevesebb alkönyvtárat számol meg, mint amennyi valójában van. Az egyetlen helyzet, amikor tényleg helyesen működik az, amikor nincs mit megszámolnia. „Ügyesen” megoldottuk tehát egy speciális eset kezelését, és elrontottuk az összes többiét.
De mi is tulajdonképpen a baj? Ahogy az előbb említettük, a `wc` az újsorkarakterek alapján „tájékozódik”. A `-n` kapcsolóval viszont éppen a lezáró újsor kiírását tiltottuk meg az `echo`-nak! A hiba tehát röviden, hogy a `-n` kapcsolót vagy kizárólag különleges esetben kellene használnunk, vagy egy teljesen más megoldást kellene alkalmaznunk.
Nyilván az olvasó is érzi, hogy mindkét fenti nehézség kezelése egy „mi történik akkor, ha” jellegű kérdés programszerkezeti megvalósítását igényli. Kicsit tényszerűbben: feltételes utasításokat kell beépítenünk a programba.

Feltételes utasítások héjprogramokban

Héjprogramokban egy feltételes programblokk szerkezete nagyjából a következő:

```
if logikai_kifejezés
then
  a feltétel teljesülése esetén érvényes
  ➥parancsok
else
  a hamis ág
fi
```

Bővebb magyarázatot talán csak a „logikai kifejezés” használata igényel. Héjprogramokban a legtöbb ilyen kifejezést a `test` parancs segítségével fogalmazzuk meg. Ez a parancs mindenféle vizsgálatot (numerikus és nem numerikus összehasonlításokat, fájlok vagy könyvtárak létezésének vagy tulajdonságainak vizsgálatát) képes elvégezni, és a feltétel teljesülését a visszatérési értékén keresztül jelzi. Ha a logikai kifejezés igaz, akkor a visszatérési érték nulla, ha nem, akkor egy.
Fontos kihangsúlyozni, hogy visszatérési értéke minden Linux alatt futó programnak (beleértve a héjprogramokat is) van, tehát elvileg bármilyen parancssor használható „logikai kifejezésként”. Az, hogy az esetek többségében mégis a `test` parancsot alkalmazzuk, pusztán annak köszönhető, hogy ezt a parancsot kifejezetten erre találták ki, az összes többit pedig kimondottan másra. Az is lényeges, hogy a logikai kifejezésekre támaszkodó programszerkezeti elemek igaznak kizárólag a nulla visszatérési értéket tekintik. (A logikai hamis értéknek bármely nem nulla érték megfelel.)

1. táblázat

egyenlő	`-eq`
nem egyenlő	`-ne`
kisebb	`-lt`
nagyobb	`-gt`
kisebb vagy egyenlő	`-le`
nagyobb vagy egyenlő	`-ge`

Numerikus vizsgálatok

A Unix operációs rendszernek ma van néhány – finoman fogalmazva – a számítástechnika hőskorára emlékeztető szolgáltatása. Ilyenek a `test` parancs numerikus összehasonlításra szolgáló kapcsolói is, amiket az *1. táblázatban* foglaltam össze. Nézzünk egy egyszerű példát:

```
if test $szam -eq 1
then
...
```

Ennek a feltételes szerkezetnek akkor hajtódik végre az igaz ága, ha a `szam` nevű változó tartalma egy. Ügyeljünk rá, hogy a `test` parancs a változó tartalmára és nem a nevére kíváncsi, tehát a `$` használata kötelező. Szintén fontos mozzanat, hogy a numerikus műveletek csak egész számokkal működnek. Gyakori, hogy ugyanezt a szerkezetet egy másik, valamivel talán áttekinthetőbb formában fogalmazzuk meg:

```
if [ $szam -eq 1 ]
then
...
```

2. táblázat

`-z` *karakterlánc*	A karakterlánc hossza nulla.
`-n` *karakterlánc*	A karakterlánc hossza nem nulla.
`-e` név	A megadott fájl létezik.
`-f` név	A megadott fájl létezik és egyszerű fájl.
`-d` név	A megadott könyvtár létezik.
`-r` név	A fájl létezik és olvasható.
`-w` név	A fájl létezik és írható.
`-x` név	A fájl létezik és végrehajtható.

```
1: #!/bin/sh
2: if test -d $1
3: then
4:    lista=`ls -l $1 | grep ^d`
5:    echo "$lista"
6:    if [ `echo "$lista" | wc -w` -eq 0 ]
7:    then
8:       darab=0
9:    else
10:      darab=`echo "$lista" | wc -l`
11:   fi
12:   echo "Összesen" $darab "alkönyvtár"
13: else
14:   echo "A megadott könyvtár nem létezik!"
15:   exit 1
16: fi
```

Itt mindössze annyi a változás, hogy a `test` parancsot szögletes zárójelekkel helyettesítettük. Ez valóban csak helyettesítés, tehát ugyanannak a dolognak egy másik írásmódjáról van szó, semmi többről. Ügyeljünk rá, hogy a szögletes zárójeleket mindkét oldalon minden egyébtől legalább egy szóköznek kell elválasztania, ellenkező esetben misztikus, a tényleges hibára nem igazán utaló hibaüzenetet kapunk!

Nem numerikus vizsgálatok

A `test`-nek számos egyéb hasznos szolgáltatása is van. Fájlokat, könyvtárakat és karakterláncokat is képes kezelni, illetve a különböző részfeltételeket logikai ÉS illetve VAGY műveletekkel kombinálhatjuk is. Ezekről bővebben a `test` súgóoldalán olvashatunk, de néhány fontosabbat a *2. táblázatban* is felsoroltam. Számunkra első közelítésben csak a `-d` kapcsoló fontos, ami egy könyvtár létezését vizsgálja.

A javított változat

Programunk javított változatában. Mint korábban említettem, két feltételes utasításra is szükségünk lesz. Az egyikkel rögtön a program elején azt vizsgáljuk meg, hogy a parancssori értékként megadott könyvtár egyáltalán létezik-e. Ha nem, akkor a további műveletekkel nyilván kár próbálkozni.
A másik döntési szerkezetben azt kell megvizsgálnunk, hogy volt-e legalább egy alkönyvtár. Ha nem, akkor a nulla darabszámot határozott módon kell beállítanunk, mert mint láttuk, az `echo -n` általános használata nem megfelelő.
A kód egyik lehetséges megvalósítása az *1. listában* látható.
A megadott könyvtár létezését a 2., míg a lista hosszát a 6. sorban vizsgáljuk meg. Ha nincs a megadott névnek megfelelő könyvtár, hibaüzenetet küldünk, majd 1-es visszatérési értékkel

```
1: #!/bin/sh
2: if [ -d $1 ]
3: then
4:    if [ -r $1 -a -x $1 ]
5:    then
6:       lista=`ls -l $1 | grep ^d`
7:       echo "$lista"
8:       if [ -z "$lista" ]
9:       then
10:         darab=0
11:      else
12:         darab=`echo "$lista" | wc -l`
13:      fi
14:      echo "Összesen" $darab "alkönyvtár"
15:   else
16:      echo "A megadott könyvtár nem
               ➥listázható!"
17:      exit 2
18:   fi
19: else
20:   echo "A megadott könyvtár nem létezik!"
21:   exit 1
22: fi
```

leállunk (15. sor). (A nullától különböző értékkel arra utalunk, hogy a leállás valamilyen hibaállapot miatt történt.)
A 6. sorban a `test` parancs által vizsgált értéket parancsbehelyettesítéssel álltjuk elő. A lista hosszának vizsgálatával kapcsolatban azt használjuk ki, hogyha üres, akkor a benne található szavak (`-w`) száma biztosan nulla. Ugyanez a sorokra (`-l`) vagy a karakterekre (`-c`) nem igaz, mivel az egyetlen újsorkarakter is egy sornak, és egyben egyetlen karakternek számít. Ezen a ponton természetesen számos más megoldást is használhattunk volna. A legegyszerűbb azonban az, ha felfedezzük, hogy a `test` parancsnak eleve van egy erre szolgáló kapcsolója, a `-z`. Ezzel a 6. sor egyszerűen a következő alakot ölti:

```
if [ -z "$lista" ]
```

Az olvashatóság vizsgálata

Még mindig akad egy dolog, ami eddig elkerülte a figyelmünket. Linux alatt ahhoz, hogy egy könyvtár tartalmát az `ls` paranccsal megjeleníthessük, legalább olvasási (`r`) és végrehajtási (`x`) jogosultsággal kell hozzá rendelkeznünk. Elképzelhető tehát, hogy az értékként megadott könyvtár létezik ugyan, de nem látunk bele.
Ennek a feltételnek a vizsgálatához egy újabb `if...then...else` szerkezetet kell beszúrnunk az eddigi igaz ágba (lásd *2. listánkon*).
A 4. sorban a logikai ÉS (`-a` kapcsoló) segítségével kombináltuk az olvasási (`-r`) és a végrehajtási (`-x`) jogosultság vizsgálatát. Hiba esetén a 17. sorban a 2-es visszatérési értékkel jelezzük a feladat természetét.

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

A fájlrendszer feladata (13. rész)

Elérkeztünk következő nagy témakörünkhöz, a fájlrendszerhez. Ez nagyon bonyolult rendszer, aminek sok feladatot kell megoldania, ezért a mostani részben többnyire csak az alapfogalmakat járjuk körbe.

A fájlok és könyvtárak fogalma senkitől sem állhat távol, hiszen a fájlkezelés az operációs rendszereknek az a feladata, amit minden felhasználó először tanul meg. Ha azonban pontosan meg kellene határoznunk, hogy mik is azok a fájlok, illetve mi a feladata az operációs rendszer fájlkezelő „alrendszerének” (a továbbiakban: fájlrendszer), aligha adhatnánk pontos választ, ha csupán a felhasználó szemszögéből közelítenénk hozzá. Ezért első ízben ezt a témakört egy kissé „elvont” formában tárgyaljuk. Erre azért van szükség, hogyha majd a megvalósítást tárgyaljuk, érthetőbbek legyenek azok a szempontok, amelyeket egy jelenlegi korszerű operációs rendszer tervezésekor figyelembe kell venni.

A fájlrendszer tehát olyan valami, aminek igazából egyetlen feladata van: az adattárolás. Tekintsünk rá úgy, mint egy olyan dologra, amibe ha beleírunk valamilyen adatot, akkor azt megőrzi (elméletileg végtelen ideig, a rendszerösszeomlástól és egyéb katasztrófáktól függetlenül), és kérésre bármikor visszakaphatjuk tőle. Ezt úgy is megfogalmazhatjuk, hogy a beírt adat bármikor visszakereshető.

Miért van szükség adatot tárolni képes eszközre? Három különböző okból – ezek közül az első minden felhasználó számára ismert. Amikor egy folyamat befejezi futását, visszaadja az általa használt memóriarészt. Az itt tárolt adat így örökre elvész. Bizonyos alkalmazásoknak azonban szüksége lehet rá, hogy a memóriájuk tartalma ne vesszen a feledés homályába, hanem a futtatásuk után is megmaradjon. Gondoljunk csak az adatbázisrendszerekre: teljesen használhatatlan lenne egy olyan megoldás, aminél a rendszer minden újraindítása után az összes adatot újból be kellene táplálni.

A másik gond, ami megköveteli a fájlrendszer meglétét, az, hogy a számítógép memóriája véges (ugyanis határt szab neki a virtuális címtér). Ezért előfordulhat, hogy az operációs rendszer nem tud annyi memóriát adni egy-egy folyamatnak, amiben az egész általa használt adathalmaz elfér. Például egy bankszámlákat nyilvántartó rendszerben lehetetlen minden számlát és átutalási kérelmet a memóriában tartani.

A harmadik (ám nem kevésbé fontos) nehézség az, hogy ez idáig nem volt lehetőség arra, hogy több folyamat egy időben ugyanazzal az adattal dolgozhasson. Visszatérve a banki nyilvántartós példához: ha egy folyamat az összes bankszámla adatait a memóriában tartotta, ahhoz más nem férhetett hozzá, mivel a memóriája a többi folyamat számára elérhetetlen.

A fájlrendszer azonban lehetővé teszi a folyamatoktól teljesen független adattárolást. Ez azt jelenti, hogy az adatok megőrzését a legcsekélyebb mértékben sem befolyásolhatja a folyamatok indulása és befejeződése. (Például ha valamelyik folyamat létrehoz egy állományt, akkor az egészen addig ott is marad, amíg valaki le nem törli. A fájlrendszer számára azonban teljesen mindegy, hogy melyik állományt melyik folyamat hozta létre, e szempontból nem lehet megkülönböztetni a fájlokat egymástól.)

Ahhoz, hogy a fent leírtakat biztosítani tudjuk, a fájlrendszernek a következő három dolgot kell tudnia: először is meg kell oldania, hogy az adat ne veszhessen el, azután is megmaradjon, hogy a folyamat befejezte a futását vagy egy összeomlás miatt újra kellett indítani a rendszert. Fel kell készülnie arra is, hogy az itt tárolt adat mérete lehet, hogy igen nagy lesz. A harmadik és egyben utolsó feladat: semmiféle korlátozás se legyen arra nézve, hogy egy adatot egyszerre hány folyamat érhet el.

Az első feladatot egyszerűen megoldhatjuk olyan módon, hogy az adatot például merevlemezen tároljuk. A merevlemez azonban buta eszköz, saját maga nem képes a tárolt adatok rendszerezésére. Ez az operációs rendszer feladata, ami tárolási egységeket, úgynevezett fájlokat vezet be, és a kezelésükről is gondoskodik. Az operációs rendszernek azt a részét, ami az állományokat kezeli, gondoskodik azok hozzáférhetőségéről és védelméről, fájlrendszernek nevezzük. (A továbbiakban a „fájl” és az „állomány” kifejezések ugyanazt a fogalmat takarják).

A memóriakezelés esetében a felhasználók elől a legtöbb folyamat rejtve maradt. A felhasználók számára érdektelen volt, hogy a virtuális memória éppen lapozott-e vagy szakaszolt. A fájlrendszer esetében más a helyzet, ezzel a felhasználó már közvetlenül „érintkezik”. Ezért a fájlrendszereket két különböző szempont alapján szokás tanulmányozni. Az első eset, amikor a felhasználó szemszögéből vizsgálódunk. Ilyenkor az számít, hogy miként végezhetünk különböző műveleteket a fájlrendszerrel, milyen szabályokat kell betartanunk az állományok névválasztásánál, milyen tulajdonságokkal bírnak az állományok stb. Ezek olyan dolgok, amelyeket a felhasználóknak ismerniük kell, és minden rendszerben másmilyenek.

A másik vizsgálati szempont a megvalósítás. Ebben az esetben azt elemezzük, hogy az állományok miként vannak tárolva a lemezen, hogyan tarthatjuk nyilván az üres blokkokat, milyen módon épülnek fel a könyvtári bejegyzések stb.

Minket természetesen az utóbbi szempont érdekel, ehhez azonban szükség van arra, hogyha csak felületesen is, de „felhasználói szemmel” is megnézzük a fájlrendszer szolgáltatásait. A következő fogalmak lehet, hogy számos olvasó számára már ismertek lesznek, mindenesetre fontosnak tartjuk, hogy kitérjünk rájuk, mivel a megvalósítás tárgyalásakor gyakran elő fognak bukkani.

Fájlnév, fájlszerkezet, fájltípus

A fájlok azért vannak, hogy a felhasználók elől el legyenek rejtve a számukra érdektelen részletek, például az, hogy az adataik fizikailag hol is tárolódnak. Inkább egy elvont eszközt adunk a kezébe, amit könnyen megjegyezhető névvel láthat el, és egyszerűen végezhet vele különböző műveleteket.

A fájlnévre vonatkozó megkötések minden operációs rendszerben mások, de a legtöbb rendszerben lehetőség nyílik a fájlnév kibővítésére, úgynevezett kiterjesztés hozzáadásával

(amelyet . – ponttal – elválasztva írhatunk a fájlnévhez). Léteznek olyan rendszerek, amelyekben komoly jelentőséggel bírnak ezek a kiterjesztések, de akad olyan is, amiben kizárólag csak a felhasználó „tájékoztatására” vezették be, ez ugyanis utal a fájl tartalmára. (A DOS/Windows rendszerekben például a futtatható bináris programok csak ***.EXE*** kiterjesztéssel bírhatnak, különben nem indíthatók el. A Unix esetében a futtatható fájloknak nincsen kiterjesztésük, de ez csak a „szokás hatalma”, valójában semmi sem tiltja, hogy a futtatható binárisoknak kiterjesztést adjunk.)

A fájlszerkezet azt jelenti, hogy az állományban lévő tartalom milyen szerkezetbe rendeződik. Ebből háromféle létezik; az első a legáltalánosabb, a Unix és a Windows rendszerek is ezt használják. Itt a struktúra maga a strukturálatlanság. Az operációs rendszer számára minden állomány csupán bájtok sorozata, egyáltalán nem foglalkozik annak mélyebb „jelentéstartalmával”. Ez ugyan kezdetleges megoldásnak tűnhet, de valójában ez biztosítja a legnagyobb rugalmasságot, hiszen teljesen a felhasználóra van bízva, hogy az állományokban mit és milyen módon tárol.

Ez az elv ma már teljesen hétköznapinak számít, de létezett olyan operációs rendszer (például a CP/M egy ősi változata), ami gyökeresen eltért ettől a szemlélettől. Ebben minden állománynak jól meghatározott szerkezete volt, mégpedig rögzített méretű (128 bájtos) rekordok sorozataként tekintett a fájlokra, sőt maguknak a rekordoknak is előre meghatározott felépítésük volt. Ez egy kicsit furcsának tűnhet, de valójában akkoriban nagyon célszerű volt egy olyan gépen, ami lyukkártyákkal és sornyomtatókkal dolgozott.

A rekordszerkezet ma sem halt ki teljesen, csak továbbfejlődött. A rekordok hosszának már nem kell állandónak lennie, és az egész faszerkezetbe szervezhető. Egyetlen megkötés, hogy – az adatbázisokhoz hasonlóan – minden rekordnak tartalmaznia kell egy kulcsmezőt. Fontos, hogy a rekordok fájlon belüli elhelyezkedését is az operációs rendszer intézi, tehát a felhasználónak nincs beleszólása, hogy melyik rekord hova kerüljön. Az ehhez hasonló rekordszerkezettel elsősorban adatfeldolgozásra használt nagygépes rendszereknél találkozhatunk. Például a VMS is támogat valami ehhez hasonlót.

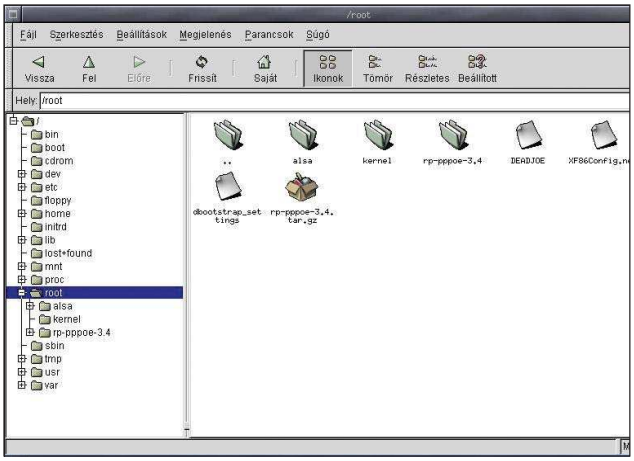
A fájlszerkezet tehát arra ad választ, hogy az adott állományban az adat milyen szerkezetben van tárolva. Azt azonban, hogy az állományban milyen jellegű adatra lelhetünk, a fájltípus mondja meg, amit szintén több különböző csoportra bonthatunk.

A fájltípusok első csoportja az egyszerű felhasználói állományok, amik a felhasználók (illetve az alkalmazások) adatait tartalmazzák. A második csoportba a rendszerfájlok tartoznak, amik a fájlrendszer számára hasznos adatot hordoznak. Ilyenek (általában) a könyvtárak is, amik a felhasználóknak „mutatott” hiearchikus szerkezetet teszik lehetővé. A Unixok egy másik fájltípust is támogatnak, mégpedig az eszközfájlt. Valójában ez két különböző típus gyűjtőneve: a karakterspecifikusé, amelyek a karakteres eszközök B-K műveleteit „vezetik ki” a fájlrendszerhez, és a blokkspecifikusé, amik a blokkos (lemezes) eszközök kezelésére szolgálnak.

Az eszköz- és rendszerállományokra csak a megvalósítás tárgyalásakor térnénk ki részletesebben, de az egyszerű felhasználói állományokról most kell szólnunk. Ezt a csoportot további két osztályra bonthatjuk: azokra, amelyek tartalma a felhasználó számára érthető, és azokra, amelyeké nem érthető. Kicsit szakszerűbben megfogalmazva: a szövegesekre és a binárisokra.

A szöveges állományokra az a jellemző, hogy tartalmukat mindenféle átalakítás nélkül kiírathatjuk a képernyőre vagy átküldhetjük a nyomtatónak. Fontos, hogy a szöveges állományok minden rendszerben egy kicsit másképp épülnek fel. Például a DOS esetében minden fájl végén egy úgynevezett „kocsi vissza” jel volt, de a rendszerek között különbség lehet például az „új sor” jelekben is. (A rendszer szöveges állományokhoz való „viszonyában” is komoly különbségek vannak. Például a Unix nagy hangsúlyt fektet a szöveges állományok magas szintű kezelhetőségére, míg egy DOS parancssorban csak a legegyszerűbb feladatokat végezhetjük el.)

A bináris állományok tartalma azonban az ember számára értelmezhetetlen, általában ugyanis mindig valamilyen belső szerkezetet tartalmaznak. A bináris állományok egyik különleges fajtája a végrehajtható bináris fájl.



A végrehajtható bináris programokat az operációs rendszer olvassa be, betölti a memóriába és elindítja őket. Ehhez az szükséges, hogy az adott fájl megfelelő felépítésű legyen.

A végrehajtható binárisok felépítése operációs rendszerenként különböző lehet, de abban megegyeznek, hogy az első pár bájt azonosításra szolgál. Például az MS-DOS esetében az EXE-k mindig „MZ”-vel, Windowsnál „NE”-vel kezdődnek. (Igazából a windowsos programok egy kis DOS-os programmal kezdődnek, ami nem csinál mást, mint kiír egy hibaüzenetet, és kilép. A windowsos program csak ezután következik.) A régi unixos binárisokban (amelyek ***a.out*** néven híresültek el) pedig egy úgynevezett mágikus szám található a fájl elején. Ezekre azért van szükség, hogy egyrészt az operációs rendszer felismerhesse a bináris programokat, másrészt a felhasználót is védi, nehogy véletlenül nem bináris programot akarjon futtatni. Minden operációs rendszernek fel kell tehát ismernie futtatható állományait. Egyes operációs rendszerek azonban más típusú állományok felismerésére is képesek. Például a Windows-rendszerek a fájlok kiterjesztéséből következtetnek a típusára, és ha egy állományra kétszer kattintunk, önműködően elindul a hozzárendelt alkalmazás. (Ám a Windows kutyafüle a TOPS-20 nevű operációs rendszerhez képest. Ha elindítottunk egy alkalmazást, az operációs rendszer ellenőrizte, hogy a bináris létrehozásának időpontja régebbi-e, mint a forrásban történt utolsó változás. Ha igen, akkor újból lefordította az adott programot.)

Felmerül a kérdés, hogy mennyire célszerű, ha egy operációs rendszer ilyen mértékben foglalkozik a fájltípusokkal. A kezdő felhasználók számára biztosan nagy segítség, mivel kisebb az esélye, hogy valami hibát követnek el (például rossz alkalmazással nyitnak meg egy állományt). A tapasztalt felhasználókat azonban idegesítheti, mivel sokkal bonyolultabban tehet meg olyan dolgokat, amelyek eltérnek a „hétköznapi” műveletektől.

Fájlelérés

A fájlszerkezet és a fájltípus mellett fontos fogalom még a fájlelérés. Ez tulajdonképpen azt határozza meg, hogy milyen módon olvashatunk (illetve írhatunk) az állományokból.
Az első operációs rendszerek fájlelérése szekvenciális volt, azaz a bájtokat (vagy rekordokat) kizárólag egymás után olvashattuk ki a fájlból. Lehet, hogy nekünk csak az 5. bájtra volt szükségünk, de akkor is előbb ki kellett olvasnunk az első négyet. A fájl első bájtjára azonban bármikor visszaugorhattunk, tehát újraolvasásra mindig volt lehetőségünk.
Ezekben az ősi rendszerekben azért nem nyílt lehetőség a fájlon belüli pozícionálásra, mert az adatok mágnesszalagon voltak tárolva, amelyek csak lineárisan írhatók és olvashatók. A mágneslemezek megjelenésével viszont ez a helyzet megváltozott, itt már csupán egyetlen meghatározott blokk beolvasására, illetve újraírására volt lehetőség. Ezért az operációs rendszerekben is megjelentek a véletlen elérésű fájlok. A véletlen elérésű állományok esetében létezik egy úgynevezett `SEEK` művelet, ennek segítségével megmondhatjuk, hogy melyik pozíciótól kezdődően szeretnénk olvasni az adott fájl tartalmát.

Tulajdonságok

Azt már tudjuk, hogy minden állománynak van neve és adattartalma. A fájlrendszer azonban mást is tárol az adott állományokról, például a létrehozásának a dátumát, a tulajdonost, a jogosultságait (ez mondja meg, hogy az adott állománnyal ki mit csinálhat); de léteznek olyan rendszerek is, amelyekben jelszót is rendelhetünk a fájlhoz. Ezeket az adatokat nevezzük az állomány tulajdonságainak (attribútum). Minél több tulajdonság adható meg egy fájlnak, annál több lehetőség rejlik a fájlrendszerben. (Például a Unixban a védelmet elég kevés tulajdonság írja le, csupán írási, olvasási és futtatási jogosultság adható. Más rendszerekben, például a Windows NT-ben vagy a VMS-ben ennél finomabb „hangolásra” is lehetőség nyílik.) Viszont az is igaz, hogy minél több a tulajdonság, annál bonyolultabb a fájlkezelés.

Könyvtárszerkezet

Ma már mindenki számára teljesen természetes, hogy állományainkat úgynevezett könyvtárakba szervezhetjük. Ezáltal lehetőségünk nyílik adataink rendszerezésére, így eszményi esetben könnyedén megtalálhatjuk a keresett adatot.
A könyvtárak azonban nem voltak mindig ennyire alapvetőek. A régi CP/M operációs rendszernél csupán egyetlen könyvtár létezett, ami az összes állományt tartalmazta. Ez az elgondolás nem lehetett célravezető egy többfelhasználós rendszer esetében, főleg, ha a felhasználók megegyező nevű állományokat akartak létrehozni. Ezután bevezették, hogy minden felhasználóhoz külön könyvtárat rendeltek. Ám azon felhasználók számára, akik nagyszámú állománnyal rendelkeztek, ez se volt túl célszerű megoldás. Végül létrejött a ma is használt hiearchikus szerkezet, tehát a könyvtárak maguk is tartalmazhatnak további könyvtárakat, amelyek szintén további könyvtárakat rejthetnek magukban és így tovább.
A faszerkezetű könyvtárszerkezetnél megjelenik az útvonal fogalma, azaz innentől kezdve az állományoknak már nemcsak nevük, hanem „helyük” is van a fájlrendszerben. Az útvonal megadására hagyományosan két mód kínálkozik: az első az úgynevezett abszolút útvonal, amikor is a hierarchia legtetején álló (gyökér) könyvtárhoz viszonyítva adjuk meg egy állomány helyét (például */home/user/alma.txt*). A másik az úgynevezett relatív címzés, ahol a munkakönyvtárhoz viszonyítunk, amit a felhasználó jelöl ki.
Sok rendszerben a könyvtárak maguk is állományok, amelyek a bennük található fájlok könyvtári bejegyzéseit tartalmazzák (lásd később).

Megvalósítás: a fájlok

A fájl tehát egy elvont adattárolási módszer. Annak nyilvántartása, hogy egy állomány tartalma pontosan hol is helyezkedik el a merevlemezen, a fájlrendszer feladata. Ennek módja is teljesen rendszerfüggő – a továbbiakban megnézünk pár gyakrabban előforduló megoldást.
Ezek közül a legegyszerűbb a folytonos helyfoglalás, ami a régi nagygépes rendszerekben volt elterjedt (még most akad olyan hely, ahol használják). Itt minden állomány kizárólag egymást követő blokkok sorozatán tárolható. Ha a lemez 1 kilobájtos blokkokba szervezett, akkor például egy 100 kilobájt méretű állományt csak 100 egymás mellett elhelyezkedő blokkon tárolhatunk.
A módszernek két behozhatatlan előnye van a többi módszerrel szemben. Egyrészt hihetetlenül könnyű megvalósítani, hiszen minden állományhoz egy kezdőértéket kell rendelni, ami megmondja, hogy hányadik blokktól kezdődik az adott fájl. Másrészt ez a valaha létezett leghatékonyabb fájlrendszer, mivel az olvasófejet csak egyszer kell pozícionálnunk, utána folyamatosan olvashatjuk az adatokat, egészen a fájl végéig.
Sajnos e két jó tulajdonság mellé két rossz is társul, de ezek az előnyök mellett teljesen eltörpülnek. Ahhoz, hogy ez megvalósítható legyen, minden állománynak létezik egy legnagyobb mérete, aminek átlépésére nincs lehetőség. (Azokban a bizonyos nagygépes rendszerekben ezt a legnagyobb méretet minden állomány létrehozásakor meg kellett adni. A fájl mérete természetesen kevesebb is lehetett, de több soha.) Másik üneprontó tulajdonsága, hogy rendkívül hajlamos a töredezésre. Töredezettség (fragmentáció) alatt azt értjük, amikor a gyakran használt blokkok nem egybefüggő részt alkotnak, hanem össze-vissza találhatók a lemezen, kisebb-nagyobb szabad blokkokkal szétválasztva. Ez azért baj, mert ilyenkor összes szabad blokkot nem használhatjuk fel, magyarul a lemez kihasználtsága romlik. Ennek orvoslásaként a fájlrendszerre kell ereszteni egy úgynevezett töredezettségmentesítő programot, ami úgy próbálja „tologatni” a gyakorta használt blokkokat, hogy közöttük a lehető legkevesebb lyuk legyen. Ez a folyamat rendkívül erőforrásigényes. Ha egy fájlrendszer hajlamos a töredezésre, akkor vagy gyakran kell töredezettségmentesíteni (ami a felhasználóktól veszi el a kapacitást), vagy jelentős szabad területről kell lemondanunk.
A láncolt listás helyfoglalás sokkal jobb megoldásnak tűnik, mivel (ahogy a nevéből is sejthető) a fájl által lefoglalt lemezblokkok láncolt listába szerveződnek. (A láncolt lista olyan tárolási forma, amiben a listában szereplő minden elem „tudja”, hogy ki a következő. Így egy tag eléréshez csak a lista első elemének helyét kell ismernünk, utána a láncon elemenként végigmenve megtalálhatjuk a keresett tagot. Ennek különleges formája a kétszeresen láncolt lista, ahol minden elem az előtte állót is ismeri.) Ezt úgy kell elképzelnünk, hogy minden blokk első pár bájtja azt mondja meg, hogy melyik blokk a következő. A fájlok nyilvántartásához itt is elég a kezdőérték ismerete, ami az első blokk címét tartalmazza.
Az ilyen módon megvalósuló fájlrendszereknél nincs szükség töredezettségmentesítésre, mivel minden üres blokkot fel tudunk használni. Ezért viszont komoly árat kell fizetnünk: az egész iszonyatosan lassú. Ha csupán az utolsó blokkhoz szeretnénk hozzáférni, akkor is be kell olvasni az összeset. Ez rendkívül időigényes feladat, mivel itt a fájl által használt blokkok nem

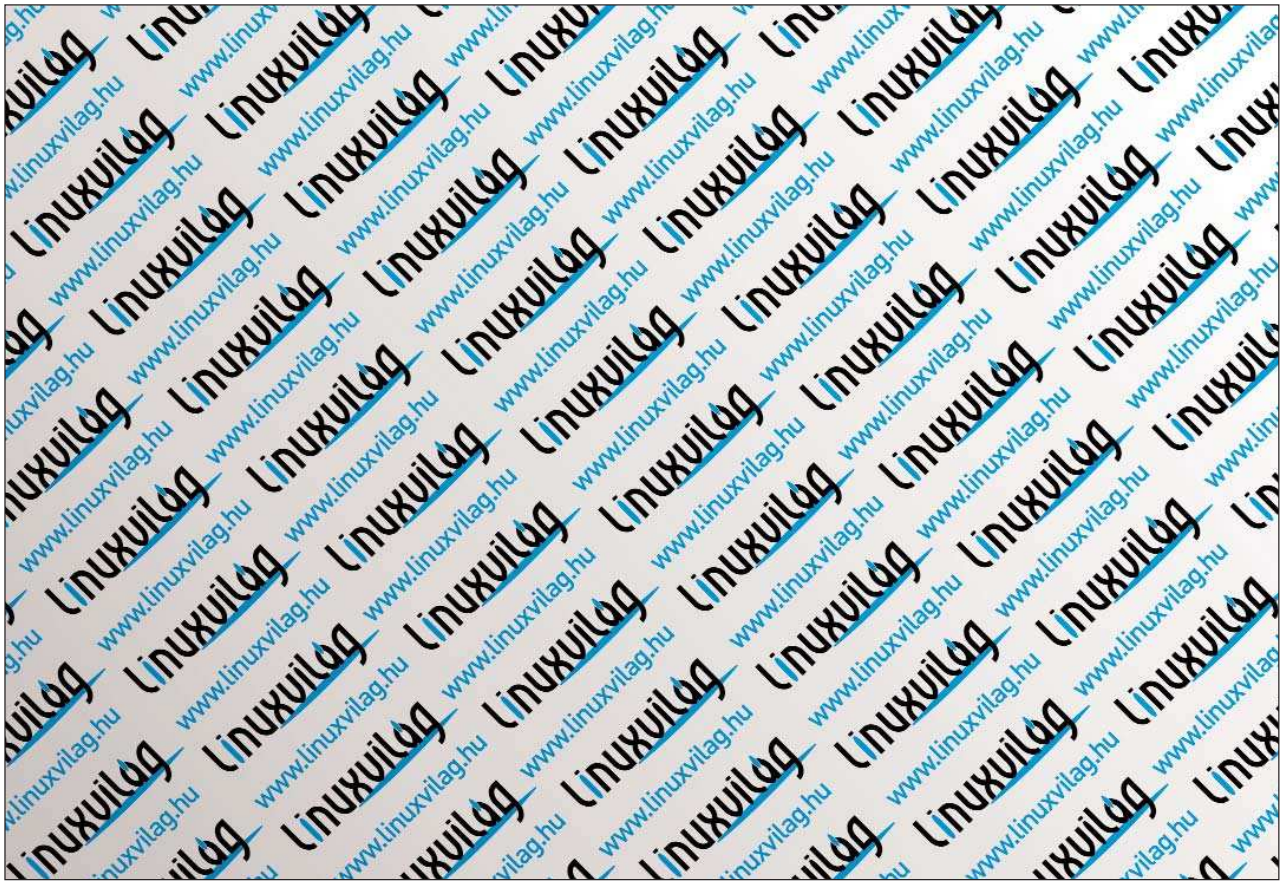
egymás mellett, hanem a lemezen szétszórva találhatók. Így minden blokk beolvasása előtt szükség van a fej pozícionálására. A másik lassító tényező az, hogy nincs lehetőség az egész blokk területét felhasználni, mivel az első pár bájt egy mutató, ami a következő blokkot jelöli meg. Ennek következtében a blokkban tárolható „értékes" adat mérete nem lesz a kettő valamely hatványa. (A lemezblokkok mérete mindig a kettő valamelyik hatványa. Ezért a programok egyszerre általában mindig kettőhatvánnyi méretű adatokat olvasnak be, illetve írnak ki. Mivel a blokk egész mérete nem használható fel, valószínűsíthető, hogy pár bájt „kimarad", így szükség lesz egy újabb blokk beolvasására, illetve kiírására.)

Szerencsére a láncolt listás megoldás hátrányainak búcsút inthetünk, ha a mutatókat (a következő blokk címét tartalmazó bájtokat) nem a blokkokban, hanem egy táblázatba szedjük össze. Ezt a táblázatot indexnek nevezzük, és a memóriában tároljuk, így annak elérési ideje nem lesz számottevő. Ennek köszönhetően hasznos adat tárolására nemcsak a teljes blokkot használhatjuk, hanem a véletlen elérés is sokkal gyorsabb lesz. Ha ugyanis nem az elejétől kezdve akarunk olvasni az állományból, nem kell az összes, a kívánt kezdőpozíció előtti blokkot beolvasni, hanem az indexből kikeresve egyből megkaphatjuk a megfelelő blokk címét.

Ez a módszer jól működött az asztali számítógépeken, ahol pár megabájtos merevlemezekkel kellett dolgozni. Az MS-DOS FAT nevű fájlrendszere is ezt használta. Idővel azonban megjelentek a többszáz, majd a többezer megabájtos merevlemezek is, és a memóriában tárolt táblázat mérete hirtelen megugrott. Például egy 500 MB-os merevlemeznél is már 2 MB-tal kellett számolni. (Ez az MS-DOS esetében megoldhatatlan lett volna, mivel ő közvetlenül csak 640 kilobájt memóriát tudott kezelni. Ezért úgy oldotta meg, hogy nagy lemezeknél nagy blokkméretet – 32 kilobájt – használt, ami rettenetes pazarlásnak számított, különösen sok kis állomány esetében.)

A fájlok megvalósításának másik megközelítése a fájlleíró (inode) használata, amit a Unix-rendszerek is alkalmaznak. Minden állomány rendelkezik egy ilyen fájlleíróval, amelyben egyrészt megtalálhatók a tulajdonságok, és néhány, a fájl által használt blokk. Ha a fájl elég kicsi (tehát ha kevés blokkot használ), akkor az összes blokk címe megtalálható a fájlleíróban. Ha nem, akkor egy másik blokkban folytatódik a felsorolás. Ezt a blokkot egyszeresen közvetett blokknak nevezzük, a címe szintén a fájlleíróban van. Ha az állomány esetleg nagyon nagy, akkor egy kétszeresen közvetett blokk használatára is lehetőség nyílik (ennek helye szintén a fájlleíróból nyerhető ki). Ez azonban már nem a fájl által használt blokkok, hanem további egyszeresen közvetett blokkok címét tartalmazza (a legtöbb Unix fájlrendszere a háromszorosan közvetett blokkok használatát is megengedi). A következő hónapban a Unix könyvtárainak taglalásával innen folytatjuk, továbbá szó lesz a lemezterület-kezelésről is. Olyan kérdéseket érintünk, hogy például miként tudjuk nyilvántartani üres blokkjainkat, illetve milyen módszerekkel tudjuk növelni a fájlrendszer sebességét.

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



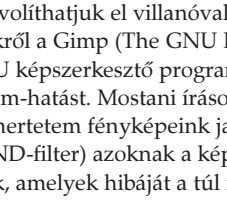
A fényképek kontrasztjának javítása Gimppel

Hogyan korrigáljuk fényképeink túl sötét tartományait anélkül, hogy a világos részek kifakulnának?

A legutóbbi cikkemben (Linuxvilág, 2003. március) azt mutattam be, hogyan távolíthatjuk el villanóval készült pillanatfelvételeinkről a Gimp (The GNU Image Manipulation Program, azaz GNU képszerkesztő program) segítségével a bosszantó vörösszem-hatást. Mostani írásomban a Gimp újabb hasznos fogását ismertetem fényképeink javítására: a digitális átmeneti szűrőt (ND-filter) azoknak a képeknek a rendbehozatalára használhatjuk, amelyek hibáját a túl nagy kontrasztú jelenetek fényképezése okozza.

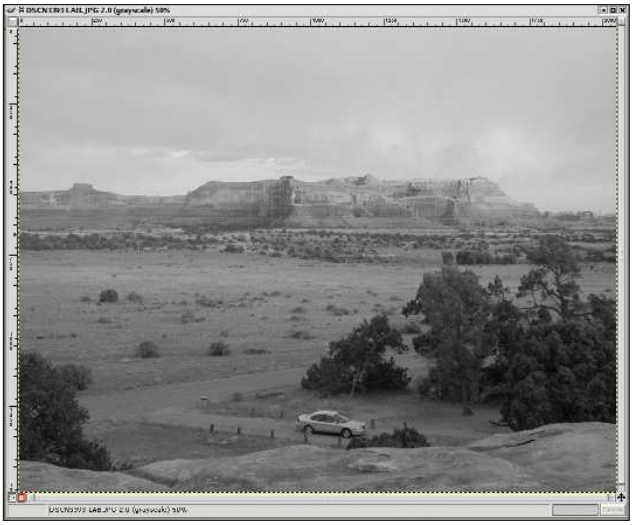
Az emberi szem igen figyelemre méltó képességekkel bíró „képérzékelő" berendezés. Képes olyan képek érzékelésére, amelyek igen széles dinamikatartománnyal (fényerősség-különbséggel) rendelkeznek, és alkalmas a részletek megkülönböztetésére mind a világos, mind pedig a halványabb, árnyékos részeken belül. A fényképezésben a fényességet gyakran a rekesznyílással mérik, ahol minden egyes érték a fény felét, illetve kétszeresét jelenti a szomszédos értéknek. Az emberi szem képes megkülönböztetni egy olyan látvány képrészleteit, amelyben a fényesség a legsötétebb és legvilágosabb rész között 14 ilyen egység. A filmek és a digitális fényérzékelők messze nem ilyen ügyesek. A diapozitív jellemzően 5–6 egységet kezel. Az alsó határnál sötétebb részek teljesen sötétek lesznek, míg a felső határértéknél világosabb részletek fehér színű résszé olvadnak össze. A negatív filmek a 9–10 egységükkel egy kicsit jobban teljesítenek, és néhány csúcskategóriás digitális fényképezőgép (DSLR) egy leheletnyivel még ennél is többre képes. A digitális fényképezőgépek 6–9 egység kezelésével ebből a szempontból a képfelfogó eszközök versenyének hátsó mezőnyében foglalnak helyet; a pillanatnyi érték függ a digitális rögzítési eljárás színmélységétől, az érzékelő méretétől és néhány más tényezőtől.

Képzettebb fotósok, ismervén a gépük korlátozott felbontóképességét, különböző eszközökkel próbálják meg csökkenteni a fényképezendő jelenet dinamikatartományát. Tehetik ezt az árnyékos részek világosításával (vaku, világítás, reflektor használatával) vagy a túl fényes részek sötétítésével, például olyan különleges szűrőkkel, mint amilyenek az átmeneti szűrők. Az *1. képen* egy ilyen szűrő látható. Ez egy olyan tartozék, amit az objektív elejére rögzíthetünk, s egy teljesen átlátszó és egy sötétszürke tartománnyal rendelkezik, ezeket folyamatos átmenet választja el egymástól. A szűrő sötét része a szűrő erősségétől függően 1, 2 vagy még több egységgel csökkenti az átengedett fény erősségét. Amikor a fényképezőgépet egy nagykontrasztú kép fotózására készítjük elő (például egy naplementére), akkor a szűrőt úgy kell beállítani, hogy a sötét része fedje le a kép világos tartományát (például az eget), az átlátszó rész pedig a kép többi részét (például a látóhatár alatti részt). Ezután a fotós a sötétebb rész alapján megmérheti a megvilágítást. Amennyiben a szűrő elhelyezése megfelelő volt, a fénymérő is helyes értéket mért – és a fényképész nem mulasztotta el lekopogni a várt eredményt, valamint egy kis imát is elmondott, miközben egy csipetnyi sót a bal válla fölött maga mögé szórt – a teljes kép megfelelő megvilágítású lesz.



1. kép Egy átmeneti szűrő

2. kép Egy utahi naplemente fényképe a Gimp ablakába töltve



3. kép A LAB-felbontás bővítménye használat közben

A legtöbb hétköznapi fotós nem foglalkozik azzal, hogy átmeneti szűrőt cipeljen magával, és ilyen esetekben azt használja. Ilyenkor az egyetlen működő megoldás az, ha engedményt

teszünk. A legtöbb önműködő rendszer az arany középutat választja, és hagyja elveszni a fényességtartomány két szélén lévő részleteket. Ha úgy döntünk, hogy kézzel állítjuk be az expozíciót, követhetjük a diát használó fotósok gyakorta emlegetett ökölszabályát: a kép szempontjából fontos világos részre állítsuk be az értékeket. Sokszor utólag még helyre lehet állítani a sötét tartományok részleteit, de ha a világos rész kiégett, onnan már nem tudunk megmenteni semmit. Azt se feledjük, hogy a szabály a ***fontos*** részt említi. Ha naplementét fényképezünk, bizonyára meg szeretnénk őrizni a napfény által beragyogott felhők mintázatát és részleteit, de ha a képünk egy mezőn álló jávorszarvast ábrázolna napnyugtakor, akkor esetleg a szarvas prémjének a részletei a fontosabbak, a felhők pedig legyenek olyanok, ahogy éppen sikerül.

Bár az ilyen megvilágítású képek összes részlete nem állítható vissza, mégis gyakori, hogy egy kis trükkel a fényes és sötét tartományokban megbúvó képrészletek jelentős része megmenthető. A hagyományos folyadékos képelőhívó eljárás ezt a folyamatot fényvisszatartásnak és kiégetésnek (dodging and burning) nevezi. Amikor a negatívról papírképet készítünk, a papír egy része jobban vagy kevésbé kerül megvilágításra a többinél, megtartva a fényes részleteket vagy előhozva a sötét tartományok különbségeit. Ezek a fortélyok azonban eddig csak a sötétkamra szerelmesei számára voltak hozzáférhetők. Most a Gimp segítségével bárki megteheti ugyanezt, mégpedig sokkal egyszerűbben. Ezt az alábbi példán keresztül szeretném bemutatni: a *2. képen* a utahi naplemente látható a Gimp egyik ablakába töltve. Követve a bölcsek szavát, a képet a felhőkre és a fényes sziklákra állítottam be, és hagytam, hogy az előtér sötét maradjon. A LAB-felbontó bővítmény segítségével ezt az RGB-képet felbonthatom a LAB alkotóelemeire. Ezek közül az L-csatorna a képen előforduló összes fényességértéket (luminance) hordozza. Ahogy az a *3. képen* is látható, a kép előterében lévő fák jelentősen részletgazdagságok, amit a kép eredeti nézete szinte teljesen elrejt a szemünk elől. De hogyan tudnám úgy előcsalogatni ezeket a részleteket, hogy közben a sziklák és felhők gyönyörű színei és részletgazdagsága ne változzék? Ennek a sötét tartománynak a megmentése kicsit olyan, mintha az átmeneti szűrő használatának a digitális megfelelőjéről lenne szó. Ugyanannak a képnek két változatát egyesítem, ahol az egyes változatok a sötét, illetve világos részre lettek egyszerűsítve. Mivel ez a módszer a Gimp réteg- és rétegmaszkkezelő képességét aknázza ki, először ezek működését kell megértenünk.

Rétegek és rétegmaszkok

A Gimpben kezelt összes kép egy vagy több réteget alkothat. Egy kép első betöltődésekor az alapértelmezett háttérréteget foglalja el, ahogy a *4. képen* is látható. További rétegeket hozhatunk létre a háttérréteg fölé. Ezek a felsőbb rétegek is tetszőleges képet tartalmazhatnak. Gyakori eset, hogy itt ugyanannak a képnek különböző változatait akarjuk tárolni. Ezt a legkönnyebben úgy érhetjük el, hogy másolatot készítünk az eredeti képről, mint az *5. képen* is látszik, ahol épp a háttérréteget kettőzöm meg. Minden rétegen a többitől függetlenül hajthatunk végre műveleteket. A *6. képen* az látható, amint éppen a felső rétegen hajtok végre szintbeállítást a kép világosítása érdekében.

Az egyes rétegek különböző módokon egyesíthetők az eredő kép létrehozásához, hasonlóan ahhoz, mintha a felső rétegen keresztül néznénk az alsót. Az egyik lehetőség, hogy a felső rétegek vagy egy részük átlátszóságát változtatjuk. Egy réteg átlátszósága a teljesen átlátszótól (0%) az átláthatatlanig terjed (100%), és a kettő között tetszőleges értéket állíthatunk be. Arra

4. kép Az alapértelmezett háttérréteg

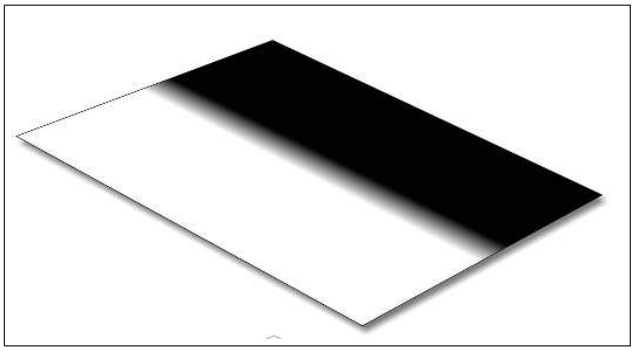
5. kép Egy réteg megkettőzése

6. kép A kép világosítása a szintbeállítás segítségével

is lehetőségünk nyílik, hogy a réteg egyes tartományai különböző átlátszóságértékkel rendelkezzenek. Ezt ismét többféle módon érhetjük el, de a legrugalmasabb módszer a rétegmaszk használata. Rétegmaszkot bármikor hozzáadhatunk a réteghez, ami a réteg tulajdonsága lesz. Ez a réteggel azonos méretű szürkeárnyalatos kép. A rétegmaszk a réteg átlátszóságát olyan módon változtatja meg, hogy annak értéke minden képpont esetén a maszk adott pixelének az értékével egyezik meg.

A maszk fekete pontja a réteg teljes átlátszóságát eredményezi a megfelelő ponton, míg a fehér teljes fedést eredményez. E két szélső érték közötti értékek a megfelelő átlátszósági százalékértéket rendelik a réteg adott pixeléhez.

És ezzel még csak a lehetőségek kezdeténél tartunk. A *7. képen* egy olyan rétegmaszk látható, ami jól használható képünk



7. kép Egy rétegmaszk

8. kép Színátmenet használata

9. kép A kép rétegeinek összeolvasztása a kimenet számára

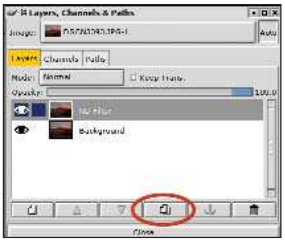
felső rétegéhez. A maszk létrehozásakor egy olyan átmenettel töltöttem fel, aminek a hatására a felső réteg átlátszósága a két szélső érték között fokozatosan változik, ahogy a *8. képen* látható. Végül a kép a *9. képen* mutatott módon egyesíthető a kimenet számára.

A rétegmaszk csak egy, a rétegek számos lehetséges tulajdonsága közül. Más tulajdonságok – mint például a keverési mód (blend mode) – azt határozzák meg, hogy az egyes rétegek milyen módon legyenek egyesítve az alattuk fekvő rétegekkel. A rétegek működésének további vizsgálatába fektetett energia bőven megtérül a Gimp képszerkesztő lehetőségeinek bővülésében.

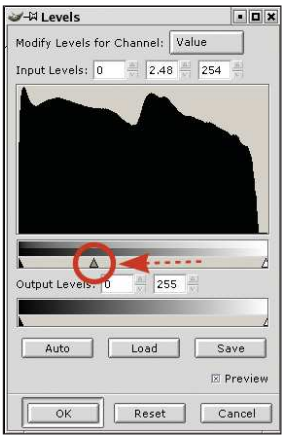
A részletek

Most, hogy felvérteztük magunkat a rétegek és maszkok kezelésének képességével, vizsgáljuk meg közelebbről az eljárást.

A Gimp legtöbb menüpontját a kép ablakán végrehajtott jobb egérkattintással érhetjük el. A következőkben ezt a műveletet JK-val rövidítem. Ha egy alkalmazandó Gimp-tevékenységet szeretnék leírni, zárójelek közé tett menüpontsorozatot vagy billentyűkombinációkat olvashatunk majd. Például egy kép megnyitásakor a (JK>File>Open) formát használom, ami annyit jelent, hogy a jobb egérgombbal kattintunk a kép ablakán, kiválasztjuk a ***File*** menüpontot, majd a megjelenő menüből az ***Open***-t. Ha célszerűbbnek tűnik a billentyűkombinációs megoldás, listát közlök azokról a billentyűkről, amiket meg kell nyomnunk. Például a kép másolására szolgáló (CTRL-C) jelentése: nyomjuk le és tartsuk nyomva a CONTROL billentyűt, majd nyomjuk meg a C-t.



10. kép A rétegek párbeszédablaka



11. kép Szintállítás használata a kép világosítására

Nyissuk meg a ***Layers*** (rétegek) párbeszédablakot az eredeti képen (CTRL-L). A párbeszédablakban kattintsunk a jobb gombbal a háttérréteg nevén (***Background***), és válasszuk a ***Duplicate*** (megkettőz) parancsot, vagy nyomjuk meg a *10. képen* bekarikázott gombot. Kattintsunk kétszer a létrehozott rétegen, majd adjuk neki az ***ND Filter*** (ND-szűrő) nevet. Ez a lépés nem feltétlenül szükséges, de felesleges kavarodástól kímélhetjük meg magunkat vele, így ha fél év múlva újra megnyitjuk a lépet, tudni fogjuk, hogy melyik réteg mit tartalmaz, és mit hogyan csináltunk. Pillanatnyilag a ***Layers*** párbeszédablaknak a 10. képen látható képet kell mutatnia. Még mindig ugyanebben a párbeszédablakban válasszuk ki az ***ND Filter*** réteget. Váltsunk át a kép ablakába, és végezzük el azokat a műveleteket, amelyekkel a fényes vagy árnyékos tartományok részletgazdagságát növelhetjük. Erre jól használhatók a szintekkel (***RC>Image>Colors>Levels***), illetve a görbékkel (***RC>Image>Colors>Curves***) kapcsolatos műveletek, de a fényerősség/kontraszt állítását (***RC>Image>Colors>Brightness and Contrast***), a megvilágítottság javítóeszközeit (dodge/burn) vagy bármi mást is használhatunk – amit alkalmasnak tartunk rá.

Mivel a másolat rétegén dolgozunk, nem kell azzal törődnünk, hogy mi történik a kép jó felével, hagyjuk nyugodtan, hogy túl sötét vagy világos legyen. Összpontosítsunk arra, hogy a javítandó tartomány megfelelő legyen. Ebben az esetben én a szinteket (***Levels***) módosítottam (a középső csúszkát a *11. képen* látható módon egy kicsit balra mozgattam), ezzel világosítottam az egész képet, míg az előtér képe a kívánt módon meg nem jelent. Ezen a ponton a 6. képen látható helyzetben kell lennünk: látható a kivilágosított kép, ami elfedi az alatta lévő eredetit.

A következő feladatunk egy rétegmaszk létrehozása, ami felfedi az alul lévő kép felső részét. A ***Layers*** párbeszédablakban a jobb gombbal kattintsunk az ***ND Filter*** rétegen, és válasszuk ki az ***Add Layer Mask*** (rétegmaszk hozzáadása) menüpontot. Az ***Add Mask Options*** (maszk beállítása) párbeszédablakon

válasszuk a ***White*** (fehér) színt (ez a teljes átlátszatlanság), és kattintsunk az ***OK*** gombra. Végül a ***Layers*** ablakon kattintsunk a ***Background*** réteg mögött lévő szemre, amivel a háttérkép láthatóságát kapcsoljuk ki. ***Layers*** párbeszéd ablakunknak ezen a ponton a *12. képen* látható állapotban kell lennie, egy kis fehér rétegmaszkikonnal az ***ND Filter*** rétegikonja mögött.

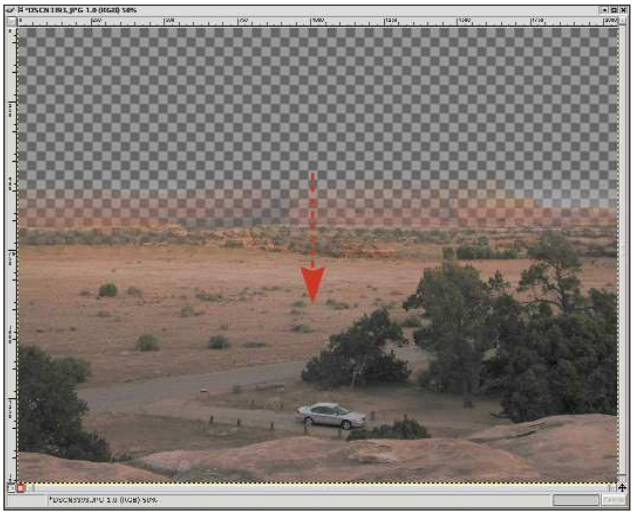
12. kép A háttérréteg láthatóságának kikapcsolása

Váltsunk át a Gimp fő eszközdobozának ablakára, és válasszuk a *13. képen* bekarikázott színátmenetes kitöltés eszközt (gradient fill tool). Menjünk vissza a kép ablakára, és az egérrel húzzunk egy vonalat olyan szögben és irányban, ahogyan a felső és alsó réteget el szeretnénk választani (a lenti réteg a vonal kezdete, a fenti pedig a vége). A húzott vonal hossza azt fogja megadni, hogy milyen fokozatos, illetve hirtelen lesz az átmenet az előálló képösszetételen. Egy kis gyakorlást igényel, hogy ráérezzünk, de általában egy rövid vonásra van szükség az átmeneti sávon keresztül (ami például a látóhatár lehet). Mivel a háttér láthatóságát kikapcsoltuk, az eredmény azonnal láthatóvá válik, ahogy azt a *14. kép* is mutatja; a kép átmeneti sávon túli része el fog tűnni. Ha nem sikerült jó helyre rakni az osztóvonalat, az egérrel egyszerűen húzzunk egy új vonást: az új átmenet azonnal megjelenik a régi helyén. Ha nem világos a mozzanat, nézzünk a 7. képen lévő átmenetre. A Layers párbeszédablakon lévő rétegmaszkikonon meg kell jelennie a most létrehozott átmenet kicsinyített másának. Emlékezzünk rá, hogy a színátmenetes kitöltés a felső réteg átlátszóságát fogja befolyásolni. A fehér nem átlátszó, a fekete teljesen az, a kettő között lévő színek pedig valamilyen mértékű áttetszőséget jelentenek.

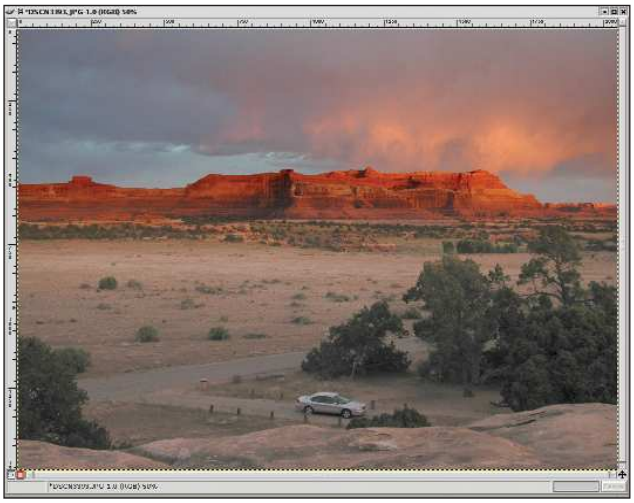
13. kép Kitöltés színátmenettel

Ideje szemrevételeznünk munkánk eredményét: kattintsunk a háttérréteg ***Background*** neve után lévő szemre a ***Layers*** ablakon, visszakapcsolva ezzel a réteg láthatóságát. És íme – az egyesített képünk, aminek mindkét tartománya megfelelően exponált. Most kattintsunk az ***ND Filter*** réteg mögött lévő szemre, és kapcsolgassuk ki-be a digitális ND-szűrő hatásának láthatóságát. Nem rossz, ugye? A *15. kép* a végső eredményt mutatja.

Ennek a megközelítési módnak az az igazi szépsége, hogy eredeti képünk érintetlenül fekszik a háttérrétegen. A második réteg világosított képe sem igényel valami hajszálpontos tartománykijelölést, hogy csak a megfelelő részt érje a hatás, és (ebben az esetben legalábbis) a rétegmaszk létrehozása is könnyen végrehajthatónak bizonyult. A teljes rugalmasság kedvéért a képet a Gimp saját xcf formátumában menthetjük, ezáltal a teljes rétegszerkezet is megmarad. Lesz még rá lehetőségünk, hogy később további módosításokat végezzünk a képen, méghozzá abban a biztos tudatban, hogy az eredeti kép sértetlen és könnyen hozzáférhető a háttérrétegen arra az esetre, ha bármikor törölni szeretnénk a végrehajtott módosításokat. Amikor készen vagyunk, a képet valamelyik népszerű képformátumba (TIFF vagy JPEG) exportálhatjuk, és ezzel a 9. képen látható módon fog megjelenni az eredmény.



14. kép Színátmenetes kitöltés alkalmazása



15. kép A végeredmény

A rétegmaszk működésének ismeretében elgondolkodhatunk olyan eljárásokon, amelyekkel sokkal összetettebb átmenetek is megvalósíthatók a fényes és árnyékos képrészek között (például egy égnek meredő sötét szikla esetén). A rétegmaszkon ugyanazokat a műveleteket hajthatjuk végre, mint bármilyen más szürkeárnyalatos képen, ezzel tetszőleges módon keverhetjük a felső és alsó rétegek tartalmát.

Remélem, hasznosnak bizonyul a bemutatott eljárás, és sok nyári napfényes felvételt sikerül megmenteni vele. Az egyik talán éppen egy görögországi utazás emléke lesz…

Linux Journal 2003. április, 108. szám

Eric Jeschke (eric@redskiesatnight.com)
Az Indiana Egyetemen számítástechnikából szerzett PhD fokozat birtokosa. Hawaii-on él feleségével, gyermekeivel és túlsúlyos macskájával. Szabadidejét legszívesebben a családjával, szabadtéri kalandokkal, fotózással és a Linuxszal tölti.

A Plone beállítása

A Plone kinézetét tetszésünk szerint formálhatjuk CMF-vezérlők, saját felületek és más Zope alapú eszközök segítségével.

A tartalomkezelő rendszerek (CMS) piaca hatalmas iramban növekszik, ám ezen nincs mit csodálkozni. Néhány évvel ezelőtt az igazgatók eltűnődtek volna azon, hogy egyáltalán készítsenek-e honlapot a cégüknek. Manapság már nem az a kérdés, hogy felállítsunk-e weblapot, hanem hogy mi módon szervezzük meg a felügyelettel foglalkozó emberek munkáját, illetve milyen rendszerben tároljuk az adatokat. A jó CMS mindezt könnyen kezelhetővé teszi, hiszen gondoskodik a felhasználókról, a csoportokról, a jogosultságokról és az időzíthető közzétételről.

Aki tapasztalatot gyűjtött a létfontosságú programrendszerek kezelése terén, jól tudja, hogy a programok ritkán (vagy inkább sohasem) kezelnek mindent, amire szükségünk van. A saját tartalomkezelő rendszereiket árusító vagy bérbe adó cégek (mint például a Vignette és a Documentum) ezt ki is használják, és tanácsadói, illetve támogatási díjakat szednek vásárlóiktól. A CMS-vásárlók a lehető legnagyobb részt szeretnék a saját kezükben tartani, egyrészt, hogy elkerülhessék a tanácsadói díjak kifizetését, másrészt, hogy mindennapjaikban nagyobb szabadságot élvezhessenek.

Nem meglepő, hogy a nyílt forrású tartalomkezelő megoldások lehetővé teszik, sőt kifejezetten javasolják felhasználóiknak a CMS-program átalakítását. A rendszer módosítása sajnos igen gyakran egyet jelent a forráskód átírásával, amire nincs mindenki felkészülve, vagy megfelelő képzettség híján nem mer nekifutni. Hatékony, rugalmas, ugyanakkor nem programozók által is könnyen módosítható CMS-rendszert készíteni meglehetősen nehéz, komoly kihívást jelentő feladat, olyan, ami valószínűleg számos CMS-forgalmazó idejét leköti majd az elkövetkezendő esztendőkben.

Az egyik egyre népszerűbb nyílt forrású CMS, a Plone, ami sok programozás nélkül is lehetővé teszi a weblap kinézetének testreszabását. A Plone nem a levegőben lóg, hanem a Zope's Content Management Framework (CMF) rendszerére épül, ami tulajdonképpen a tartalomkezelő rendszerek készítésére tervezett API-k gyűjteménye. Mint azt a múlt hónapban láthattuk, a Plone segítségével egyszerűen készíthetünk olyan weblapokat, amelyek olyan fejlett képességeket tartalmaznak, mint az eseménynaptár, a hírújság vagy a keresőmotor.

Vajon mi a helyzet, ha mindezt meg szeretnénk változtatni? Mit tegyünk, ha a Plone alapértelmezett kinézete nem felel meg az ízlésünknek? Szerencsére a Plone számos módon és több különböző szinten módosítható. Ebben a hónapban megnézünk néhány, a Plone módosításához használt módszert, s mindeközben sokat megtudhatunk a Zope CMF-rendszeréről, ami kitűnő átmenet lesz jövő havi CMF-körutunkhoz.

Alapvető változtatások

A Plone testreszabásának egyik legalapvetőbb módszere, ha módosítjuk a kinézetét. Ehhez a hely gazdájaként kell bejelentkeznünk. Feltételezve, hogy a szabályos, alapértelmezett módszer szerint telepítettük a Plone-t, ezt két módon tehetjük meg. Az egyszerűbb lehetőség, ha gazdaként bejelentkezünk a Plone-ba, és megadjuk azt a jelszót és felhasználói nevet, amit webalapú Zope kezelőfelületen használnánk (ZMI). A Plone normál esetben örökli a jelszavakat és szabályokat az őt körülvevő környezettől, így a Plone-ba ezekkel a hitelesítő adatokkal is be tudunk jelentkezni.

Noha a Plone-ba bejelentkeztünk, sajnos a Zope-oldal többi része nem tekint bennünket bejelentkezettnek. Hogy elkerüljük ezt a nehézséget, és hogy egyszerre tudjuk kezelni a Zope és a Plone rendszereket, jelentkezzünk be az oldalunkra a */manage* címen keresztül is: ha például a honlapunk a *www.example.com* címen fut, akkor a *www.example.com/manage* címen próbálhatunk meg rendszergazdaként bejelentkezni.

Miután rendszergazdaként beléptünk a Plone-oldalra, egy menüsort láthatunk a képernyőn a főmenü alatt, a „you are here” sortól rögtön jobbra. Kattintsunk a Plone *setup* nevű menüpontjára.

Miután beléptünk a Plone-beállításokba, válaszolnunk kell néhány kérdésre, például hogy mi legyen Plone-oldalunk (vagy a portálunk, ahogy a Plone hívja) neve, illetve hogy melyik levélcímre küldje a Plone a rendszerüzeneteket. Az egyik pontban kiválaszthatjuk az alapértelmezett kinézetet, az alapértelmezett telepítésben található körülbelül tucatnyi lehetőség közül. A honlap kinézete azonnal frissül, így a kiválasztott kinézetet megtekinthetjük és meg is változtathatjuk.

Mint a neve is mutatja, az itt kiválasztott színséma csak az alapértelmezett változat. A rendszer összes felhasználója beléphet a *My Preferences* menüpont alá, ahol saját oldalának a látványát átalakíthatja. Így aztán, ha honlapunkat a legújabb Mozilla stílusúra állítjuk is, megcsontosodott ízlésű felhasználóink nyugodtan választhatnak bármi mást.

A CMF-vezérlők használata

A Plone nagy része sajnos nem állítható be belülről. Ehelyett a Zope kezelőrendszer felületét kell használnunk, úgy, mintha a Plone egyszerűen csak a CMF egyik összetevője lenne. A Plone alapértelmezett kinézetének módosításához jó néhány CMF-vezérlőt fel kell használnunk.

Ezekhez a vezérlőkhöz olyan módon jutottam el, hogy a böngészőmet a Plone-példányom telepítési helye feletti könyvtárra irányítottam. Azaz ha a Plone-oldalt mondjuk a *www.example.com/atf* címen érem el, a *www.example.com/manage* címen kapom meg a kezelőfelületet. A kezelőfelület a legfelső szint összes objektumát bemutatja, beleértve a Plone-példányunkat is. A Plone-objektumra kattintva (jelen esetben ez az atf lesz) egy hosszú objektumlistához jutunk, amiben a legtöbb elem a „portal” szóval kezdődik:

portal_catalog, portal_calendar, portal_skins, portal_membership, portal_undo

és még hosszan sorolhatnánk. A csavarkulccsal jelölt objektumok lesznek a CMF-eszközök. Ezek teszik lehetővé,

hogy Plone-példányunk egyes részeit módosítsuk. Például a ***portal_actions*** eszköz segítségével a Plone-oldal különféle részein megjelenő, különböző dobozszerű gombokat állíthatjuk be. Ide tartoznak az oldalak tetején állandóan megtalálható gombok, például a hírek (***news***) vagy a teljes körű keresés (***advanced search***), illetve azok a gombok, amelyek akkor jelennek meg, amikor a webes rendszergazda a hálózaton keresztül szeretné megváltoztatni a tartalmat. Minden műveletet hét mező határoz meg, ezek a következők:

- A ***művelet neve***, ami a dobozban a külső világ számára megjelenik.
- Az ***egyedi azonosító***.
- A ***művelet***, amit akkor kell elvégezni, amikor a felhasználó a dobozra kattint, s amelyet TALES formátumban adunk meg (a websablonokhoz szánt Zope Page Templates rendszerből), és általában egy címre mutat.
- Az (elhagyható) feltételek, amelyek szerint a gomb látható lesz. Például a ***Paste*** gombnak csak akkor kell megjelennie, ha a vágólapon érvényes adatot találunk, ezt a feltételt TALES formátumban a `folder/cb_dataValid` alakban adhatjuk meg.
- A szükséges ***jogosultság***, amelynek birtokában a felhasználó láthatja a gombot. Ha például a műveletnek ***View*** (megtekintés) jogosultsága van, akkor, aki bejelentkezett, láthatja a műveletdobozt és kattinthat is rá. Ezzel szemben, ha a művelethez ***Modify folder content*** (könyvtártartalom-módosítás) jogosultságot rendelünk, akkor csak azok a felhasználók láthatják a műveletgombot, akik a könyvtár módosítási jogosultságával rendelkeznek.
- A ***kategória***, ahová a dobozt helyezhetjük. Ez lehet a (képernyő tetején megjelenő) ***portal_tabs*** vagy ***object_tabs***.
- Végül a műveleteket a jelölőnégyzetek be-, illetve kikapcsolásával ***megjeleníthetjük*** vagy ***elrejthetjük***.

A műveletek hozzáadása, törlése és módosítása tehát igen egyszerű. De mi a helyzet, ha a Plone-oldal bal és jobb oldalán megjelenő portleteket szeretnénk kibővíteni, törölni vagy a sorrendjüket megváltoztatni? Ezeket az elemeket a Plone-rendszerben foglalatnak (slot) nevezzük. Testreszabásukhoz közvetlenül Plone-példányunk tulajdonságait kell megváltoztatnunk. Tehát rá kell kattintanunk a létrehozott Plone-példányunkra (jelen esetben ez az atf lesz), majd az oldal tetején megjelenő ***properties*** fület kell választanunk.
A ***left_slots*** és ***right_slots*** tulajdonságok határozzák meg, hogy mi jelenjen meg az oldalon. Amennyiben Plone-példányunkat mostanában telepítettük, azonnal láthatjuk, hogy az egyes foglaltok több vonalat tartalmaznak, mint ahány foglalat a képernyő tetején megjelenik. Ennek az az oka, hogy a portlet csak akkor jelenik meg, ha van mit megjelenítenie. Ha például nincsenek pillanatnyi események, a Plone meg sem jeleníti az események portletet. A ***left_slots*** harmadik sorában megnevezett portlet elsőként, másodikként vagy harmadikként jelenhet meg, attól függően, hogy az első és második tartalmaz-e megjelenítendő adatokat.
A saját oldalamon el tudtam érni, hogy az eseménylistázás a bal oldalon, míg a naptár a jobb oldalon jelenjen meg (a ***login*** portlet pedig tűnjön el). Mindehhez egyszerűen a ***left_side*** és ***right_side*** tulajdonságok meghatározásait kellett módosítanom. Az ilyesfajta módosítások könnyűek és gyorsak, és lehetővé teszik, hogy oldalunkon csak a kívánt szolgáltatások jelenjenek meg.
Végül kattintsunk a ***portal tulajdonságok*** hivatkozásra, amit Plone-példányunkon belül a ZMI jelenít meg. Ennek Plone-ikonja lesz a csavarkulcs ikon helyett, jelezve, hogy ez az eszköz egyedileg a Plone-hoz készült, és a CMF-ben általánosan nem érhető el. Rákattintva négy különböző tulajdonságlista jelenik meg (***form_properties***, ***navigation_properties***, ***navtree_properties*** és ***site_properties***), amelyek mindegyikében Plone-oldalunk valamilyen tulajdonságát változtathatjuk meg. Ha a ***site_properties***-re kattintunk, a lista furcsán ismerősnek tűnhet. Ennek az az oka, hogy a ***site_properties*** listában többet megtalálunk az eddig látott beállítások közül. Maga a Plone csak a legáltalánosabb és legszükségesebb beállításokat ismerteti; az összetettebb és haladóbb beállításokat a ZMI-n keresztül érhetjük el. Nem igazán számít például, hogy a dátumformátumot a ZMI-ből, vagy a Plone beállításoldalán állítjuk-e át; mindkét esetben az oldal azonnal átalakul – tükrözve a változásokat.

Saját bőrök

A tulajdonságmeghatározások megváltoztatásával és a ZMI használatával Plone-oldalunk tekintélyes részét átalakíthattuk. De ha igazán meg szeretnénk változtatni Plone-oldalunkat, a rendszerrel érkező lapsablonokba (ZPT) kell belenyúlnunk. Ezt könnyebb mondani, mint megtenni. Az alapértelmezett ZPT-k a következő mappákban tárolódnak a fájlrendszeren: ***$ZOPE_ROOT/lib/python/Products/CMFDefault/skins*** (ezek tartoznak a CMF tartalomhoz) és ***$ZOPE_ROOT/lib/python/Products/CMFPlone/skins/*** (a Plone-tartalomhoz). Ha ebben a könyvtárban megváltoztatjuk a bőröket, az összes Plone-példányra hatással leszünk, és nem biztos, hogy ezt szeretnénk.
A Plone szem előtt tartja ezt a lehetőséget is, ezért felajánlja, hogy a ZPT-ket a Zope objektumkönyvtárában tároljuk (ZODB), ahol pontosan ugyanúgy átszerkeszthetjük őket, mint bármely más ZPT-t. Például ZMI alól lépjünk be a ***portal_skins*** eszközbe, majd a ***portal_skins*** alatt a ***plone_templates*** könyvtárba. A ***plone_templates*** éppen olyan, mint minden hagyományos Zope-könyvtár (eltekintve eltérő ikonjától), viszont itt a ZODB belseje helyett a lemezen található fájlokat látjuk. Majdnem az összes Plone-ban látható lap ZPT-jét a ***plone_templates*** tartalmazza. A ***plone_templates*** alatti ***ui_slots*** könyvtárban található sablonok a portletek kinézetét szabályozzák.

KAPCSOLÓDÓ CÍMEK

A Plone-nal kapcsolatos legjobb forrás a ➲ http://www.plone.org. A Plone Book – bár nem teljes – a Plone telepítésével és testreszabásával kapcsolatos gazdag leírást tartalmaz.
Mivel a Plone ZPT-ben íródott, érdemes kicsit felfrissíteni ZPT-, TAL- és TALES-ismereteinket, mielőtt komolyabb módosításokba fognánk. A Zope Corporation most készülő könyve, a Zope Book (amit a közösség sok tanáccsal és észrevétellel segített) két fejezetet szentel e kapcsolódó módszerek bemutatására. A könyvet a ➲ http://www.zope.org/Documentation/Books/ZopeBook/current/ZPT.stx
és a ➲ http://www.zope.org/Documentation/Books/ZopeBook/current/AdvZPT.stx címen találjuk.
A Zope-ot és rengeteg Zope-pal kapcsolatos anyagot a Zope-közösség lapján, a ➲ http://www.zope.org webhelyen találjuk.

Amennyiben Plone rendszerünkben az oldalak tetején megjelenő fejlécet szeretnénk módosítani, rákattinthatunk a fejléc (header) ikonra. Innen egy olyan oldalra jutunk, ahol megtekinthetjük a fejléclapsablont, de nem módosíthatjuk. Ha módosítani szeretnénk a fejlécet, ki kell másolnunk egy könyvtárba, ami csak a ZODB-n belül létezik. Kattintsunk a ***Customize***-ra, és látni fogjuk, amint a cím a ZMI-be kerül, a ***portal_skins/plone_templates*** helyett a ***portal_skins/custom*** könyvtárba rakva bennünket. Ez az egyedi könyvtár az összes testreszabott sablon központi raktára, ahol az elemeket nyugodtan átszerkeszthetjük, akárcsak bármely más ZPT-oldalt a rendszeren. Mivel minden egyes Plone-példányhoz külön egyedi könyvtár tartozik, biztosak lehetünk benne, hogy az itt végzett változtatások csak az éppen módosított példányon lesznek érvényesek.

Ha nincs más dolgunk, érdemes végignézni a fejléc (header) és lábléc (footer) lapsablonokat, hogy láthassuk, milyen rengeteg beállítási munkába került (s melynek nagy része JavaScriptben készült), hogy a Plone biztosan működjön a különféle böngészők alatt. Mivel minden böngésző egy kicsit másképpen kezeli a CSS- és JavaScript-módszereket, lenyűgöző látni a Plone-fejlesztők hatalmas munkáját amivel, amennyire csak lehetséges, megpróbálták kiegyenlíteni a különbségeket.

Természetesen ez egyben azt is jelenti, hogy néhány meglepetés érhet bennünket. Apám, aki egészen mostanáig Netscape 4 alatt dolgozott, arról panaszkodott, hogy az új weboldalam folyton megfenyíti, hogy miért nem használ korszerűbb böngészőt. Mivel én már elég régóta a legfrissebb Mozilla- és Galeon-változatokat használom, még soha sem láttam ezt az üzenetet. A Web sokkal jobb hely lenne, ha minden alkalmazás ilyen okos és lelkiismeretes lenne a rendszerek közötti együttműködés terén.

Összegzés

A Plone valószínűleg a legismertebb és legnépszerűbb alkalmazás, amit a Zope CMF-rendszerével készítettek. Hatékony és könnyen beállítható. A részleteket igazán sokféle módon megváltoztathatjuk: használhatjuk a jellegzetes Plone-beállítás-képernyőket, beállíthatunk dolgokat a ZMI-n keresztül és módosíthatjuk a lapsablonokat is, ha ZODB egyedi könyvtárába másoljuk őket. Akár új, saját bőröket is adhatunk a Plone-hoz, újabb elemmel kiegészítve a terjesztéssel érkező érdekes és változatos megoldásokat.

Természetesen a Plone csak egy a Zope CMF-rendszerére épülő alkalmazások közül. Következő hónapban egy újabb elvonatkoztatási réteget fogunk visszafejteni, közvetlenül a CMF-be fogunk belekukkantani, és megvizsgáljuk, hogy milyen alkalmazásokat készíthetünk a segítségével. Mint láthattuk, megvan az oka annak, amiért a CMF olyannyira felkeltette a Zope-közösség, sőt magának a Zope vállalatnak is a figyelmét.

Linux Journal 2003. június, 110. szám

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

A Sharp Zaurus SL-C700

A Zaurus SL-5500 és az SL-5600 – ami egy módosított, erősebb processzorral, jobb akkumulátorral és mikrofonnal ellátott SL-5500 volt – sikere után a legtöbben arra számítottak, hogy a következő Zaurus legalább annyi újdonságot hoz majd, mint amennyivel első bemutatásakor az SL-5500 is szolgált. Úgy tűnik, hogy a Zaurus SL-C700 megfelel ezeknek az elvárásoknak. Az SL-C700 kisebb az elődeinél, és jóval kifinomultabbnak is tűnik. Amikor megkaptam, a jegyesem azonnal ki akarta próbálni, ami azért is meglepő, mert a korábbi modellek nem igazán keltették fel az érdeklődését. Az SL-C700 nem tökéletes, hibái leginkább a hivatalos támogatás hiányából erednek, amit a Sharp csak Japánban biztosít. A támogatás elmaradása sajnos jelentősen befolyásolja a készülék használhatóságát.

Jellemzők

A gép 400 MHz órajelű Intel XScale PXA 250 processzorral, 64 MB Flash és 32 MB RAM memóriával rendelkezik, 640×480 képpont felbontású VGA kijelzője 65 536 szín megjelenítésére alkalmas; találunk rajta IrDA és egy USB-kaput, a CompactFlash és az MMC-SD kártyák kezelésére egyaránt képes. Mindezt sztereohang, egy görgő, egy kényelmes billentyűzet és olyan kijelző egészíti ki, ami a hagyományos számítógépes fekvő és a zsebtitkároknál megszokott álló módban egyaránt használható – alighanem ez a jelenleg kapható legjobb zsebtitkár.

A zenebarátok bizonyára értékelni fogják a kiváló hangkimenetet, a sokat gépelők pedig meg lesznek elégedve a készülék méreteihez képest kiváló billentyűzettel, a grafika kedvelőit viszont a kristálytiszta képű kijelző fogja elbűvölni. Engem leginkább a szürke burkolat kellemes és minőségérzetet keltő tapintása ragadott meg. A korábbi változatok műanyag játékszernek tűntek, ennél azonban a külcsín is összhangban van az értékes belbeccsel. A csuklópánt bal oldalán található LED-ek, amelyek az akkumulátor töltéséről és az elektronikus levelek érkezéséről tájékoztatnak, illetve a kényelmesen használható, OK és Cancel gombbal kiegészülő oldalsó görgő egyaránt felhasználóbarát tervezésről tanúskodik.

Az egyetlen apróság, ami hiányozhat, a beépített mikrofon. Mivel a zsebtitkárokba kerülő mikrofonok általában nem sokat érnek, kéz nélkül is használható mikrofonos fejhallgatókat pedig bárhol vehetünk, ez talán nem is olyan nagy veszteség. Szerencsére a Sharp figyelt a vásárlók visszajelzéseire, és beépített egy belső hangszórót. Furcsának tűnhet, hogy hangszórót kapunk, mikrofont pedig nem, ám így a bekapcsolva hagyott SL-C700 különféle, akár egyedi hangjelzésekkel is értesítheti használóját, ha például elektronikus levele érkezett.

Rendelés

Az SL-C700 hivatalosan csak Japánban kapható. A Dynamism.com oldalán azonban 699 dolláros árért bárhová kérhetjük a szállítást, a helyi adókat és az illetékeket természetesen nekünk kell állnunk. A készülék hatalmas sikert aratott Japánban, a rendelések teljesítésével állandóan le vannak maradva, így ne számítsunk közeli szállítási határidőre. Rendelésem teljesítésére nekem két hónapot kellett várnom, így mire a FedEx futárja megérkezett, tulajdonképpen már nem is számítottam rá.

A Dynamism.com felkért egy, a Zaurus-közösségben jól ismert programozót, hogy az egész készüléket honosítsa angolra. Az alapértelmezett japán alkalmazások eltávolítása után tehát egyetlen japán szót sem láttam sehol – kivéve a könyvjelzőket. Teljes értékű angolsága ellenére a készüléken a szóköz és az Fn gomb között megmaradt két veszélyes billentyű, amelyekkel japán betűket írhatunk be. Ha ezeket megnyomjuk, akkor japán nyelvű módra váltunk. Az angol betűtípusok miatt az alkalmazások nem tudják megjeleníteni a japán karaktereket, így csak apró négyzetek tűnnek fel, amelyeket a japán helyesírás-ellenőrző szorgalmasan meg is jelöl pirossal.

Egy kis időre szükségem volt, míg rájöttem, hogy ezeket a gombokat kell újra megnyomni, egészen addig, amíg egy A betű meg nem jelenik a címsorban. Ha japán nyelvű módba váltunk, akkor az A betű helyett egy japán karakter jelenik meg ugyanitt.

Az új Zaurus
Gyártó: Sharp
Elérhetőség: ➲ http://www.Dynamism.com
Ár: 699 dollár

Első benyomások

Az SL-C700 billentyűzetének gombjai nagyok és puhák, alapállapotban azonban idegesítő csippantás jelez minden gombnyomást. Szerencsére a tálca hangikonjára kattintva előhívhatjuk az ***Audio Settings*** (Hangbeállítások) menüt, ahol a csippantást le lehet tiltani. Ezután a gombnyomásokat már sokkal kellemesebb halk kattanások kísérik.

Az SL-C700 kijelzőjének képminősége lenyűgöző, ilyen kiváló kijelzőt még nem láttam. A kép annyira tiszta, hogy az egyes képpontokat nem is lehet elkülönülten látni. A kijelző és a színei annyira erőteljesek, hogy háttérvilágítással akár a szabadban is kényelmesen használhatjuk. Az SL-5x00 sorozat kijelzője sem volt éppen rossz – sokkal jobb, mint például az IPAQ-é –, ám nem vette fel a versenyt például a legújabb Sony Clié LCD kijelzőkkel. Az SL-C700 kijelzője bármelyik versenytársénál jobb, ez bizonyára némi irigységgel tölti majd el az eddig oly büszke Clié-tulajdonosokat.

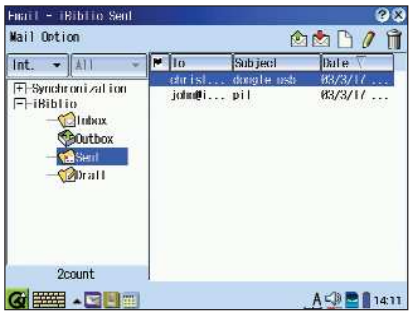
Az SL-C700 első bekapcsolásakor négy percet igényel a rendszer elindítása. Indítás közben az SL-C700 egy felbukkanó ablakban jelezte, hogy fel kell töltenem. A dobozban egy kisautó méretű töltőt találtam, japán, illetve amerikai szabvány szerinti dugóval. Átalakítóval az Európában szabványos 230 volton is remekül működött, bár a felirata szerint csak 100 volttal használható. A Dynamism.com tájékoztatása szerint feszültségátalakítóra nincs szükség. A töltő kis mérete fontos tényező, az akkumulátorral ugyan csak rövid ideig használható a készülék, ám a töltőt bárhova magunkkal vihetjük, így ez nem jelent különösebb gondot.

Bejelentkezés

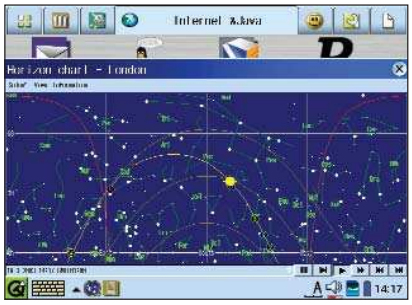
A rendszerindítás befejeztével a beállító képernyő fogad minket. A kijelző kalibrálását öt érintéssel végezhetjük, majd meg kell adnunk a helyi időt, a dátumot és az időzónát. New York vagy Tokió városát, illetve további fontos városokat alapbeállítás szerint is megtalálunk. Következő lépésként az alapértelmezett indítópult jelenik meg. A gépen alapállapotában is elég sok alkalmazást találunk. Én elsőként a személyi adatkezelő (PIM) programot próbáltam ki. Kellemetlenül érintett, hogy a naptárban és a címlistában megjelenő betűk túl kicsik voltak – hamarosan rájöttem azonban, hogy az összes alkalmazásban ez a betűtípus szerepel. A japán nyelvű kézikönyvben található ábrák alapján megállapítottam, hogy a betűk, bizony, kisebbek a kelleténél. A Dynamism.com műszaki támogatása levelemre válaszul megerősítette, hogy az angol nyelvű honosítás nem várt mellékhatása az eltérő alapértelmezett betűtípus. A hibát többen is jelezték a Dynamism.com felé, a cég most már a megfelelő betűtípussal szállítja a gépeket.

További furcsaság a címlista által alkalmazott betűrend: A, Ka, Sa és így tovább. Ez a japán ábécé szerinti rendezés, ami az angol nyelvűek számára legalábbis szokatlan. Megszűnt az SL-5x00 sorozatban megismert XML-támogatás is. Valakinek a Sharpnál volt egy olyan fantasztikus ötlete, hogy az egész világon szabványként elfogadott XML formátum helyett valamiféle sejtelmes bináris formátumban a ***~/Applications/dtm*** könyvtárban mindenféle furcsa nevek alatt tárolják az adatokat. Nekem személy szerint a régi, a ***~/Applications/Datebook*** és a ***~/Applications/Addressbook*** könyvtárban tárolt XML-fájlok tökéletesen megfeleltek, hiszen ezeket más alkalmazásokból is könnyen írni vagy olvasni lehetett. Feltelepítettem tehát az eredeti SL-5x00 Addressbook alkalmazást, ennél elég volt az ***addressbook.xml*** fájlt a ***~/Applications/Addressbook*** könyvtárba másolni. Az SL-5x00 sorozat naptára az SL-C700-ason sajnos nem működik; a beállított találkozók helyett csak egy vonalat jelenít meg. Úgy döntöttem tehát, hogy a beágyazott Korganisert fogom használni, ami a méltán népszerű asztali változat formátumát alkalmazza. Az ***addressbook.xml*** fájlt és a naptárat egyszerűen rámásoltam egy CompactFlash kártyára, majd onnan a Zaurusra. Hiába vetettem be mindent – napokon keresztül túrva a fórumokat –, sehogy nem sikerült rávennem a Zaurust arra, hogy az adatait akár Windows, akár Linux alatt összehangolja. Az asztali gép beállítása szövevényes rejtvényhez hasonlítható, talán később, ha lesz hozzá angol nyelvű leírás és több időm jut rá, újra nekilátok.

1. kép Az indítópult nagyméretű ikonjai



2. kép A kiváló elektronikuslevél-támogatás az SL-C700 fontos erőssége



3. kép A Java-alkalmazások kihasználják a kijelző kínálta lehetőségeket

Az irodai csomag

A HancomWord és a HancomSheet, a Zaurus szövegszerkesztője és táblázatkezelője elég érdekes volt az SL-5x00 sorozat tagjain. Mostanra kiforrott, gyors és könnyen használható alkalmazásokká értek az SL-C700-on, amelyek gond nélkül írják és olvassák a Microsoft formátumait is. Bár a beolvasás néha lassú egy kicsit, jó érzés, ha a dokumentumaink mindenhol velünk lehetnek.

Az SL-C700 nagyméretű, 640×480-as felbontással dolgozó kijelzője kényelmes felületet biztosít a felhasználó számára. A kijelző forgatását, valamint a 640×480-as fekvő és a 480×640-es álló mód közti átváltást tökéletesen megoldották. Ha szövegszerkesztésre használjuk, a szélesebb kép és a billentyűzet jó szolgálatot tesz, egyébként pedig a magasabb képen – a váltás egyetlen mozdulattal elvégezhető – jól áttekinthetően jeleníthetők meg adataink.

Irány a hálózat!

Az internetkapcsolat beállítása gyerekjáték. Vagyok olyan szerencsés, hogy otthon vezeték nélküli hálózatom van, így csak el kellett indítanom a hálózati beállítások megadására szolgáló alkalmazást, nevet kellett adnom a kapcsolatnak, be kell írnom a titkosító kulcsomat, majd kiválaszthatom az auto (dhcp) módot, aminél a gép önműködően hozzárendelt IP-címet kapott. Amikor beillesztettem egy vezeték nélküli CompactFlash hálózati kártyát, egy földgömb ikon jelent meg egy nagy piros keresztel. Az ikonra kattintva megtekinthettem az elérhető kapcsolatok listáját, ahol kiválasztottam a vezeték nélkülit, és ezzel be is fejeződött a beállítás. A kapcsolatlista léte egyben arra is utal, hogy az SL-C700 gond nélkül tud váltani a különböző hálózatok között – egy hordozható eszköznél ez sem hátrány.

Az SL-C700-on a Netfront böngészőt és a Qtopia levelezőprogramot találjuk. Ezek is túltesznek az SL-5x00 sorozatnál megismert elődeiken. Míg az Opera az SL-5x00 apró kijelzőjén nem tudott minden oldalt megjeleníteni, addig a Netfront az általam meglátogatott oldalakat kivétel nélkül tökéletesen kezelte.

Ha a szöveg túl kicsi vagy túl nagy, az Fn-3 és az Fn-4 gyorsbillentyűkkel könnyedén átméretezhetjük a kijelző tartalmát. Mindezt az Fn-1 és az Fn-2 gyorsbillentyűk egészítik ki, amelyekkel a kijelző fényerejét lehet növelni és csökkenteni, igazodva a környezeti fényviszonyok változásához.

A fülek segítségével egyszerre több webhelyet is megnyithatunk. A böngészőt könnyű használni, menüelemeinek és ikonjainak a jelentése könnyen érthető, beállítási lehetőséget viszont keveset találunk benne. A könnyű használat azért is előny, mert leírást csak japán nyelven találni hozzá.

A levelezőprogram beállítása és használata sem okozhat nehézséget. Meg kellett adnom a POP-kiszolgálómat, a felhasználónevemet és a jelszavamat, a kimenő leveleket kézbesítő kiszolgálót, és máris levelezhettem a Zaurusról. A válaszokat azonnal, kapcsolat nélkül is megírhatjuk. A helyi tároláshoz túlságosan nagy leveleket a program nem tölti le. Természe-

tesen, ha szükségünk van rájuk, ezeket a leveleket is letölthetjük, de mindegyik ilyen műveletet külön, kézzel kell engedélyeznünk. Én ezt hasznos szolgáltatásnak tartom, hiszen így megelőzhetjük a zsebtitkár memóriájának levélszeméttel és semmire sem használható mellékletekkel való megtöltését.

Multimédia

Az SL-C700-on hang- és mozgóképlejátszót egyaránt találunk, ezekhez egy képnézegető és egy hangfelvevő társul.
A zenelejátszó tökéletesen elboldogul az MP3-fájlokkal, illetve lejátszási listákat is összeállíthatunk vele. A kimenet minősége kiváló, a lejátszóprogram felülete pedig könnyen átlátható. A hangerőszabályzóval és a véletlenszerű lejátszás lehetőségével kiegészítve az SL-C700 szinte digitális zenegéppé változtatható.
A zenelejátszóban egy további ötletes lehetőséget találunk: a készülék kijelzőjét teljesen ki tudjuk kapcsolni, ezzel kímélhetjük az akkumulátort, de természetesen a zenét is tovább élvezhetjük.
A képnézegető apró előnézeti képekként jelenítette meg a CompactFlash kártyámra másolt képeket. Az előnézeti képekre kattintva a képeket egyenként is megnézhetjük, illetve bemutatót indíthatunk. Kedvesem teljesen el volt ragadtatva, hogy a kisméretű kijelző ellenére milyen jó minőségben láthatja viszont a szilveszter este készített fényképeket. A képek élesen és olyan gyönyörű, telt színekkel jelentek meg, hogy nehéz volt elhinni: valóban egy zsebtitkárt tartunk a kezünkben.
Amikor utazok, nem szeretek hordozható gépet cipelni magammal, viszont ha a fedélzeten játszott film unalmas, akkor nem nagyon tudok mit kezdeni magammal. Abban viszont biztos vagyok, hogy a következő alkalommal már nem kell irigykednem a mellettem ülő hordozható DVD-lejátszójára. Ha rámásolok egy DivX-fájlt egy 256 MB-os CompactFlash kártyára, majd végignézem a zsebtitkáromon, akkor biztos vagyok abban, hogy a szomszédom fog az irigységtől sárgulni.
A Doctor Z mozgóképlejátszónak köszönhetően kiváló minőségben, képkockák eldobása nélkül lehet filmeket nézni, ha azokat a támogatott DivX kodeknek megfelelő beállításokkal és az SL-C700 kijelzőjének és képességeinek megfelelő frissítési értékkel rögzítették.

Java

Az SL-C700 figyelemre méltó képessége a teljes értékű Java-támogatás. Engem érdekel a csillagászat, az idegen galaxisok, így mindig az égbolttérképeket telepítem fel elsőként. Ellátogattam a Solun webhelyre (➲ http://www.piecafe.demon.co.uk), és letöltöttem az SL-5500-hoz készített Java-változatot. Gond nélkül működik az SL-C700-on, bár a 640×480-as felbontás kiaknázása érdekében néhány apró beállítást módosítani kell. A nagy felbontásnak hála használata kényelmes, mégha a memória hiánya hamarosan rá is ébreszt bennünket: az SL-C700 sem tökéletes.

Hiányosságok

A mézeshetek elmúltával be kellett ismernem, az SL-C700 sem hiba nélkül való. Az első gondok a felbontás kapcsán jelentkeztek. Ugyan az SL-5x00 sorozattól eltérő processzorral rendelkezik, az SL-5x00-asokra készült alkalmazásoknak az SL-C700-on is futniuk kell. Sajnos némelyiket szabott méretű képernyőre készítették, így hiába áll rendelkezésükre nagyobb kijelző, nem képesek az arányos méreteződésre. A Sharp éppen emiatt tett elérhetővé egy alacsonyabb (240×320) felbontású álló módot, ami az SL-5x00 típusok kijelzőjét hivatott utánozni. Alapértelmezés szerint is elérhető, csakhogy az eltérő felbontású módok közötti váltás négy másodpercig tart, ami egy örökkévalóságnak tűnhet, ha egy alkalmazásra azonnal szükségünk van.
A Doctor Z mozgóképlejátszó hibátlanul működik, és hivatalosan is alkalmas az SL-C700-on való használatra. Az SL-C700 kiváló elektronikuslevél-támogatással rendelkezik. Kipróbáltam az ingyenes, SL-5x00 típusokra készült alkalmazásokat, és a legtöbb még nagyobb felbontás mellett is gond nélkül működött. Mindössze egy hosszú kattintás az alkalmazás ikonján, váltás a másik felbontásra – csupán ennyi kell egy próbához.
A másik bosszantó dolog a memória hiánya. 64 MB flashmemória, vagyis 64 MB tárhely, amiből 30 MB marad a felhasználónak. Már a 32 MB is soknak tűnhet egy zsebtitkár esetében, ám a Qtopia zabálja a memóriát, a felhasználónak nem sokat hagy. Ha egy zenelejátszót és egy parancssort is elindítottam, mindössze 600 KB szabad memória maradt. Még rosszabb, hogy egy tiszta rendszerindítás után mindössze 4 MB memória használható fel, ami meglehetősen szűk mozgásteret szab az alkalmazásoknak. A Java-alkalmazások futtatásakor rengeteg memóriahiányra utaló hibaüzenetet kaptam. A gép egyre jobban lelassult, majd hirtelen megjelent egy üzenet, amely néhány alkalmazás leállítására kért.
Sokkal komolyabb gondom volt a gép SD aljzatával. Nem tudom miért, de ha beillesztettem egy SD-kártyát, a gép teljesen lemerevedett, és ez az állapot csak a kártya eltávolításával szűnt meg – ekkor jött a már említett, négy percen keresztül tartó rendszerindítás. Visszsza kellett küldenem a Dynamism.com-nak.
Az SL-C700-hoz semmilyen tokot vagy kiegészítőt nem találtam. Japánban már szállingóznak az első kiegészítők, ám japán nyelvtudás hiányában még az interneten keresztül is elég bajos a rendelés.
Azt azonban be kell ismernem, hogy minden hiányossága ellenére hiányzik nekem az SL-C700-asom.
A gyári személyi adatkezelő csomag nem az igazi, a multimédiás, irodai és az internetes csomag azonban többet is tudott, mint amit egy zsebtitkártól elvártam volna. Ha bajlódtam is vele, az leginkább a hivatalos angol nyelvű támogatás hiányának tudható be. Az XML-támogatás, illetve a visszafelé együttműködési lehetőség hiánya az SL-5x00-as sorozat személyi adatkezelőjével már súlyosabb dolog, és ezen a hivatalos támogatás sem tud segíteni.
A SL-C700-fórumokon (➲ http://externe.net/zaurus/forum) talált tanácsok alapján napi rendszerességgel is használatba veszem az SL-C700-ast, ha végre visszaérkezik. A korábbi zsebtitkárom, a Zaurus SL-5500 helyét akkor fogja teljes értékűen átvenni, ha találok egy jó adatkezelő összeállítást, amely az asztali gépemmel is képes lesz összehangolni.

Előnyök

- Kiváló kijelző.
- Valódi billentyűzet.
- Irodai alkalmazások.
- Internetkapcsolat.
- Multimédiás képességek.
- Színvonalas Dynamism.com-garancia.

Hátrányok

- Kevés memória.
- A CPU gyorsítótárát a Sharp letiltotta, lelassítva ezzel a gépet.
- A gyári személyi adatkezelő alkalmazások közel használhatatlanok.
- Túl rövid akkumulátoros üzemidő.
- A kézírás-felismerés lassú és sokat hibázik.

Linux Journal 2003. június, 110. szám

Guylhem Aznar
Az LDP (➲ http://www.tldp.org) irányítója. A való életben tanácsadó, hatodéves medikus, PhD-dolgozatát informatikai témából írja. Kevéske szabadidejében a Zaurusával játszadozik.

Játszótér

Múlt hónapban a Quake különböző változatait mutattuk be, most az ennek grafikai motorjára épülő játékokkal ismerkedhetünk meg.

Ebben a hónapban megpróbálom a teljesség igénye nélkül végigpásztázni a Quake változatainak grafikai motorjaira épülő játékprogramokat. Minden Quake-változatra igaz, hogy az id Software készített hozzá hivatalos küldetéslemezeket. Ezekből a küldetéslemezekből sajnos csak egyetlen egy áll rendelkezésemre, így a lemezek hiányában – legnagyobb bánatomra – nem tudom őket bemutatni.

Hexen 2

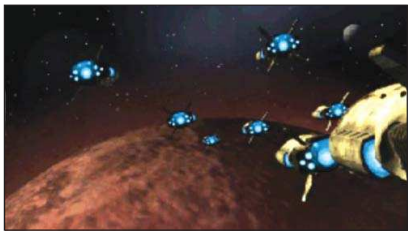
A Quake első változatának grafikai alapjára épült a Hexen 2 program, ami szintén id fejlesztés. A linuxos keresztségben a Hexen 2 átfordítása az Anvil of Thyron nevet kapta (➲ aot.linuxgames.com).
A telepítőprogram (***aot-1.2.0.x86.run***) Loki-alapokra épült (mint általában

minden linuxos játékprogram telepítője). A játék gépre varázsolásához szükség van a régi DOS-os lemezre, azt be kell fűzni, mivel a program keresni fogja a telepítéshez szükséges forrásokat. További szolgáltatás, hogy a program kezeli a Hexen 2 küldetéslemezt is (Portal of Praevus).
A telepítési folyamat végeztével a programot az `aot` parancs kiadásával lehet indítani. Itt lehet és kell is megadni az indítási értékeket.
Lényeges elem a megjelenítés beállítása, ahol a felbontás mellett meg lehet adni, hogy szeretnénk-e 3D-gyorsítást használni. Mivel a program eredeti változatának 3D része 3Dfx-re készült még Win9x alá is, így nem kell meglepődni, ha esetleg nem megy minden 3D-kártyán.
A Hexen 2 a Quake származásának köszönheti, hogy mind Windows, mind Linux alatt rendelkezik programból megvalósított, valamint GL Hexen elnevezésű 3D-s résszel. Az biztos, hogy a régi TNT-s kártyámon pár éve még nem ment. Most viszont tökéletesen fut a gépemen, a GL Hexen. Pont ezért érdemes kudarc esetén megpróbálni a programból megvalósított,

3D-gyorsítás nélküli futtatást, mert így még játszhatóvá válik a program. Azoknak, akiknek sikerül futtatniuk a programot, a képeink bemutatják, hogy milyen képvilágra számíthatnak.

Quake 2 Ground Zero

A Quake 2 egyik küldetéslemeze a Ground Zero. Sajnos csak ez a CD-m van meg. A korongot a Rogue Entertainment készítette.
Az anyag Linuxra telepítéséhez két út vezet. A múlt havi CD-mellékleten szereplő ➲ http://icculus.org oldalon is elérhető telepítőprogram támogatja a Quake 2 küldetéslemezeket.
A küldetéslemez futtatásához, mint a küldetéslemezek esetében általában szokásos, szükség van az eredeti Quake 2 korongra, valamint annak tartalmára (lásd a múlt havi leírást). A telepítéshez feltétlenül kell a küldetéslemez windowsos változata is. A windowsos CD *Data/max/Rogue* könyvtárában lévő ***video*** könyvtárnak hivatkoznia kell a ***/usr/local/games/quake2/rogue*** könyvtárra. A korong ***Data/all*** könyvtárból a ***pak0.pak*** fájlt és ***docs*** könyvtárat be kell másolni a ***/usr/local/games/quake2/rogue*** könyvtárba. A ***patch*** könyvtár a ***quake2*** kiegészítésére szolgál. Bennünket, Linux-felhasználókat csak a ***baseq2*** könyvtár érdekel belőle. A CD ***Data/patch/baseq2/maps.lst*** fájlt a ***/usr/local/games/quake2/baseq2*** könyvtárba kell másolni. A ***Data/patch/baseq2/players*** könyvtárban lévő ***crakhor***, ***cyborg***, ***female***, ***male*** könyvtárak tartalmát be kell másolni a ***/usr/local/games/quake2/baseq2/players/*** azonos nevű könyvtárakba – természetesen csak a különbséget és csak az újabb állományokat. Egészen pontosan a ***cyborg***-ból a ***disguise_i.pcx,disguise.pcx*** fájlokat, a ***female*** könyvtárból a ***disguise_i.pcx***, ***disguise.pcx***, ***rogue_b_i.pcx***, ***rogue_b.pcx***, ***rogue_r_i.pcx***, ***rogue_r.pcx*** fájlokat szükséges átmásolni. A ***male*** könyvtárból a ***cyborg***, ***female***, ***male*** könyvtárakat (ezek a CD-n a ***male*** könyvtáron belül vannak) és még a ***disguise_i.pcx***, ***disguise.pcx***, ***rogue_b_i.pcx***, ***rogue_b.pcx***, ***rogue_r_i.pcx***, ***rogue_r.pcx*** fájlokat is át kell másolni.



A játék futtatása a következőképpen történhet, a program könyvtárába lépve a következő parancsot kell kiadnuk:

```
./quake2 +set vid_ref glx
➥+set gl_driver libGL.so
➥+set game rogue
```

Esetleg a múlt hónapban közölt Quake2 parancsfájlból kiindulva:

```
#!/bin/sh
##############################
# Load Quake2 from
```

```
# /usr/local/games/quake2
##############################
cd /usr/local/games/quake2
./quake2 +set vid_ref glx
➥+set gl_driver libGL.so
➥+set game rogue
```

Kingpin Life of Crime

A játék motorját a Quake 2 adja. A fejlesztésért a Xatrix, valamint az Interplay felelős (mára a program egyik helyen

sem érhető el), nekik köszönhető, hogy a Quake 2 amúgy unalmas szögletes tereiből egy viszonylagos értelemben igazán jó és új világot alakítottak ki. Ennek ellenére nem szabad olyan megjelenítést, látvány- és élményvilágot várni tőle, mint ami a Quake 3-ban vagy az Unreal Tournamentben látható, mert arra nem képes. Mégis a maga sajátos hangulatával, beszédstílusával szórakoztató játék (megközelítőleg percenként hangzik el néhány otromba kifejezés). Az utcák egy kicsit kihaltak ahhoz képest, hogy egy bűnnel teli utcaképnek felelnek meg, viszont a kopasz, tetovált emberek lánccal az orrukban és a szintén tetovált csajok szuper módon hatnak játék közben a hangulatodra.
A sztori nagyon egyszerű: egy zsarut kell alakítanod, aki nem teljesen tökéletes, pénzért gyakorlatilag mindenre hajlandó. Az utcát uraló bandák között ügyesen lavírozva eljut a végső célig, ez: leszámolni a Kingpinnel. Hogy mi a jutalom? Természetesen pénz! Ja igen, és rengeteget lehet lövöldözni!

Telepítés

Itt is két változat lehetséges, ki-ki döntse el, hogy melyiket szeretné alkalmazni.

Az első módszer

A fájlt (***kingpin-1.20_glibc-i386linux2.0.tar.gz***) az előre létrehozott ***/usr/local/games/kingpin*** könyvtárba a `tar xfvz-kingpin-1.20_glibc-i386-linux2.0.tar.gz` paranccsal ki kell csomagolni.
A ***kingpin.conf*** fájlt át kell helyezni a ***/etc***-be. A ***libGL.so***, ***libMesaGL.so***, ***libMesaGL.so.2***, ***libMesaGL.so.2.6*** fájlt törölni kell, mivel teljesen fölöslegesek. A program futtatásához sajnos fel kell tenni a Kingpin játékot egy windowsos lemezrészre, vagy WineX segítségével fel kell telepíteni a programot a linuxos gépünkre. A telepített helyről a ***kingpin\main*** könyvtár tartalmát át kell másolni a ***/usr/local/games/kingpin/main*** könyvtárba. Az átmásolás után át kell alakítani a fájlok nevét, mert a kiterjesztések nagy- és kisbetűsek vegyesen, például van ***.TGA*** és ***.tga*** kiterjesztés is. A ***/usr/local/games/kingpin/main*** könyvtárban található a `ccase` program, amit ha `./ccase -r`-ként futtatunk, helyreállítja a dolgokat, átalakítja a fájlok nevét, kiterjesztését: minden fájlnak kisbetűs lesz a neve és kiterjesztése. Ezzel sajnos még mindig nincs vége a dolognak, mivel az összes ***libGL.so*** és ***libMesaGL.so*** fájlt töröltük, tehát létre kell hozni egy hivatkozást, ami a 3D-s kártya OpenGL-es vezérlőjére mutat. Az alábbi utasítást kell kiadni:

```
ln -s /usr/lib/libGL.so.xxx
➥"/usr/local/games/kingpin/
➥libGL.so
```

(Ebben az esetben az x a vezérlő változatszáma.)
A játék indítása a `./kingpin +set vid_ref glx +set gl_driver libGL.so` utasítással lehetséges (ha belépsz a ***/usr/local/games/kingpin*** könyvtárba). Szerintem érdemes egy parancsfájlt írni hozzá, amit ha bemásolunk a ***/usr/bin*** könyvtárba, például ***run-kingpin*** néven, máris le lehet egyszerűsíteni a folyamatot.

```
#!/bin/sh
##############################
# Load Kingpin from
# /usr/local/games/kingpin
##############################
cd /usr/local/games/kingpin
./kingpin +set vid_ref glx
➥+set gl_driver
➥libGL.so + developer 1
```

A `+ developer 1` engedélyezi a játékban a konzolt, ahova majd a különböző parancsokat vagy cheateket be lehet írni. A játék első indítása előtt rendszergazdai jogosultsággal ki kell adnunk a következő parancsokat:

```
chown root kingpin
chmod 4711 kingpin
```

Ez az érték szintén a Quake-örökségből származik, a programnak szüksége van a videoalrendszer teljes körű hozzáféréséhez.

Második módszer

Szintén az ➲ icculus.org weboldalról tölthető le a ***kingpin-1.20-x86.run***. A telepítőprogram tartalmazza azt a pluszszolgáltatást, hogy az előtelepítéshez nincs szükség sem windowsos lemezrészre, sem pedig WineX-re, mert a szükséges állományokat lemásolja a windowsos telepítőkorongról, majd a fájlokat a megfelelő formátumra alakítja át, és a telepítő által létrehozott könyvtárba másolja őket.

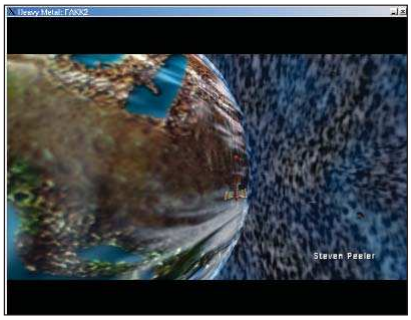
Fontos megjegyezni, hogy a Loki-alapokra épülő telepítőprogram nem tudja kezelni az `automount` vagy `supermount` segítségével önműködően befűzött CD-kötetet. A telepítés idejére, valamint a játék futtatása alkalmával is a windowsos Kingpin CD-t kézzel kell egy általunk választott helyre befűznünk (nyilván nem oda, ahova az `automount` vagy a `supermount` befűzi). Talán botorságnak tűnik a feladat, de ezzel a kis megalkuvással használható csak a telepítő és a játék is. Aki nem használja egyik önműködő befűzésre alkalmas lehetőséget sem, az természetesen nem ütközik ilyen nehézségekbe.
Végül nézzük, milyen eszközök állnak hősünk rendelkezésére az emberirtáshoz:

- ólomcső
- pisztoly
- vadászpuska
- dobtáras géppisztoly
- gránátvető
- rakétavető
- lángszóró
- gépkarabély

A játék kinézetéről és a pályákról meséljenek a képek. Ajánlom mindenki figyelmébe, hogy álljon be a vadászpuskával egy lámpa alá, majd kezdjen el forogni, mozogni! Közben érdemes figyelni, hogyan változnak a fények a puska csövén. Sajnos az árnyékkezelés annyira gyatrára sikerült, hogyha egy ház tetején, a tető peremén állunk, akkor az árnyékunk ragyogóan a levegőbe vetül, a semmire.
A CD-mellékelten megtalálható a ***kpded...*** fájl, ami a Kingpin játék kinevezett kiszolgálójaként való futtatásához szükséges, valamint ha valaki új pályákat, bővítményeket szeretne készíteni, akkor a ***kingpinsdk...*** fájlra is ráakadhat itt.

Heavy Metal F.A.K.K. a négyzeten

A játék igazi kultusszal bír. Itt minden fordítva volt és lett. Először is elkészült a Heavy Metal című képregény, amit ***Simon Bisley*** írt és öntött formába a '70-es évek végén. A Heavy Metal szürreális világa furcsa érzést keltett mindenkiben, akadtak utálói, és vannak olyanok, akik máig imádják a művet. Tizenkilenc év elteltével az alkotók újra elővették a témát. A Heavy Metal 2000 címet viselő rajzfilm főhősét az alkotók ***Julie Strain***-ről, az egykori Penthouse-modellről mintázták.
A rajzfilm egy Julie nevű hősnőről szól, akinek a húgát elrabolják az idegenek. Julie rájön, hogy az E.T.-k háta mögött egy isteni babérokra pályázó gazember, Lord Tyler áll. Ám a Szentföld véres csatamezőin vívott harcban alaposan ellátja a baját.



Történet

A csata után Julie és húga elindult az Iker Napok galaxisába, útközben maguk köré gyűjtötték a hontalanokat, hogy új otthont hozzanak létre az Éden világán. Hamar felfedezték, hogy az Éden bolygó vize gyógyító erejű, képes megőrizni az ifjúságot. A bolygó köré egy védelmi burkot, a F.A.K.K.-ot (Federation Assigned Ketogenic Killzone) állítottak fel, ami megóvja őket bármilyen, az űrből jövő támadás ellen. Az űrből egy hajó közeledik feléjük, ez a PlanetShip néven futó gépezet bolygók elpusztítására is képes; vezetője a bomlott agyú Gith.

Telepítés

A telepítőprogram 378 MB-ot fog felmásolni a gépünkre, ami nem mondható soknak; további előny, hogy a telepítést követően nem kell a CD-meghajtóban tartani a korongot, mert a játék nélküle is fut.
A cheat és egyéb kódok használatához érdemes a konzolt bekapcsolni.
A lemez a játék 1.02-es változatát tartalmazza, tehát a windowsos kiadásból a legutolsót, így nincs semmilyen frissítés hozzá, viszont az összes, a kiadásig megismert hibát javították benne. Igaz, Julie néha így is be-bedől a falba, de a hibákat nagymértékben ellensúlyozza, hogy a program még alacsony felbontás esetén is fantasztikus látványvilággal örvendeztet meg bennünket.
Külön figyelmet érdemes fordítani rá, hogyha teljes képernyős üzemmódról ablakosra váltunk, akkor az egér „kihúzható" a játékfelületről – ennek a képek készítésekor nagy hasznát vettem.
Szintén a programozókat dicséri, hogy az Unreal és Unreal grafikai motorra épülő programokkal ellentétben itt a játékon belülről lehet állítani a színmélységet.

Heretic 2

A program Quake-öröksége még az első változatból származtatható, mivel az még tisztán id-fejlesztés volt.
A következő változat viszont a Raven alkotása, igaz, ez a verzió is Quake-alapokon, egészen pontosan a Quake 2 grafikai motoron alapul.
A Quake 2-höz képest viszont harmadik személy nézetű kamerakezeléssel

vértezték fel, ami teljesen eltér a Quake 2-től. A pályák elrendezése, valamint a grafika lényegesen eltér a Quake 2 által kínált nyomasztó egyhangúságtól. Itt a városok falai, a terek, a víz, valamint a víz mozgása sokkal élethűbb. Főhősünk sem az a durrbele katona, aki csak a fegyvereire tud támaszkodni. Itt igen is léteznek varázslatok, láncsóvák és minden, ami a néhol misztikus hangulat keltéséhez, illetve fokozásához szükséges.

Telepítés

A telepítőprogram indítása, valamint a telepítés befejezése után a CD-mellékleten lévő ***heretic2-install-x86.run*** fájlt kell futtatni, mivel az eredeti játék az OpenGL alapú 3D-s kártyákat nem kezeli, mindenféle hibaüzenettel lefagy.
A frissítést követően azonban már az nVidia alapú 3D-kártya is használható. Nem kell megijedni, mert a frissítés nagyon sok fájlt eltávolít, de ezt csak és kizárólag a minél jobb megfelelőség

érdekében teszi. Egy másik lényeges elem, hogy a CD-mellékleten önmagában lévő frissítés is képes a teljes játékot a Loki lemezről telepíteni, viszont valamilyen furcsa oknál fogva ebben az esetben is fagyást tapasztaltam. Amennyiben a műveletet nem előzi meg a gyári lemezről az eredeti első változatú telepítőprogram használata, akkor sajnos a fent említett hiba mindig elő fog jönni, amint a videobeállításokban a programból megvalósított megjelenítést OpenGL-re állítjuk át. Leküzdve a telepítők által létrehozott furcsaságokat, órákra leláncolhatjuk magunkat linuxos gépünk elé.

Return to Castle Wolfenstein

A program Quake 3-alapokon nyugszik, de annak képi világát jóval felülmúlva próbál meg bennünket elkalauzolni a Wolfenstein kastélyban.

A cselekmény

Te vagy B. J. Blazkowicz, sok kitüntetéssel büszkélkedő katona, aki újoncként kerül be a Titkos Akciók Hivatalához (Office of Secret Actions – OSA). Feladatod, hogy elmenekülj, majd visszatérj a Wolfenstein kastélyba, ahol megakadályozod Heinrich Himmler titkos genetikai kísérletét. Himmler hisz abban, hogy újraélesztheti a X. század fekete hercegét, Henry Fowlert, aki szintén olyan sötét figura, mint Himmler. A genetika tudománya és az okkult erők segítségével Himmler megteremti a megállíthatatlan hadsereget.

A felhasználható fegyverek listája:

- kés,
- 45-ös Colt,
- Mauser pisztoly,
- MP40-es,
- Thomson,
- hangtompítós géppisztoly,
- kézigránát,
- lángszóró,
- páncéltörő,
- méreg.

Telepítés

A CD-mellékleten található ***wolf-linux-1.33.x86.run*** állományt kell futtatni, majd a GOTY (Game of The Year Edition) ***wolf-linux-GOTY-maps.x86.run*** bővítményt érdemes telepíteni. Legvégül a legutolsó frissítést ***wolf-linux-update-1.4.1.x86.run*** szükséges futtatni.

Az Interneten létezik egy, a teljes 1.4.1 változatot tartalmazó 100 MB feletti állomány is, én azonban megmaradtam az ➲ ftp.idsoftware.com helyről elérhető tartalomnál. A játék futtatásához szükség

lesz még egy windowsos RTCW CD-re is. Lehetőleg a GOTY-változatra, mivel azon a pakfájlok nincsenek becsomagolva, így közvetlen módon lehet őket a lemezről bemásolni a */usr/local/games/wolfenstein/main* könyvtárba. Természetesen ehhez be kell fűzni a lemezt. Az egyszerűség kedvéért az összes ***.pak*** kiterjesztésű fájlt próbáljuk meg átmásolni, és csak a régebbieket kell felülírni. Figyelem, mivel a GOTY-változat kétlemezes, mind a két korongról le kell másolni a fájlokat! Akinek csak az egylemezes változat van meg, az vagy Windows alá telepítse fel a programot, és onnan nyerje ki a szükséges pakfájlokat, vagy használjon WineX-et, amivel Linux alá is fel lehet telepíteni a programot arra az időre, amíg a pakfájlok átmásolására sort kerítünk.

A játék a csaknem kilenc évvel ezelőtti Wolfenstein utódja, de annak grafikai tudását magasan túlszárnyalja. A történet, a megjelenítés, az akciók mind-mind kiforrottak, és már az első pillanattól fogva hatással vannak a játékosra. Bár akadnak olyanok, akik a M.O.H.A.-ra esküsznek, nekik várniuk kell, amíg a linuxos átültetés befejeződik, és ugyanazt a játékélményt tudják kihozni a programból Linux alatt is, mintha a windowsos változattal játszanának. Nekem hiányzik a stratégiai rész az egyjátékos üzemmódban, a pályákon a segítséget nyújtó karakterekkel az együttműködés nulla. Elmondják a mondókájukat, majd

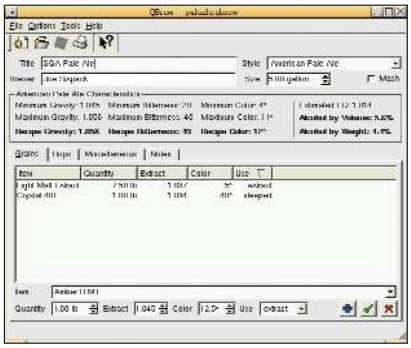
elfordulnak, és mehet az ember, amerre lát. Az Enemy Territory küldetéslemez a szövetséges erők remélhetőleg fokozott együttműködésével minden kívánságot, illetve hiányosságot pótol, és a taktikai elemek növelésével egy még izgalmasabb játékkal játszhatunk.

Kosztandinovszki Norbert
(kosztandinovszki@dialec.hu)
Linux- és játékmániás számítógépőrült.

Ügynökök védelmében

QBrew

A virtuális sör is remek, de a valódi tagadhatatlanul jobb. Miért ne lépnénk még tovább, hogy kifőzzük a saját ízesítésű sörünket? A programhoz leírás is tartozik, ami bevezet a valódi – nem

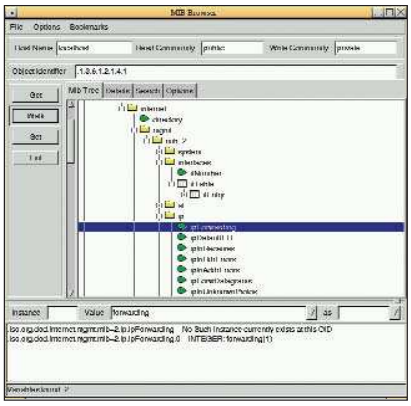


a virtuális – sörfőzés rejtelmeibe. Ha sikerült összeállítanod egy igazán jóféle világos sört, nekem is küldd el a receptjét! Futtatásához szükséges: libSM, libICE, libXext, libX11, libqt-mt, libstdc++, libm, libgcc_s, glibc, libdl, libfontconfig, libaudio, libXt, libpng, libz, libGL, libXmu, libX render, libXft, libfreetype, libpthread, libexpat.

➲ http://www.usermode.org/code.html

Mbrowse

Íme egy kellemes, egyszerűen használható és könnyen telepíthető MIB-böngésző (MIB = felügyeleti információs adatbázis). Akik SNMP-t használnak, azok tudják, hogy egy ilyen eszköz mennyivel egyszerűbbé teszi az ember életét. A Details (részletek) lap ismerteti azokat a MIB-eket, amiket alig ismersz vagy csak ritkán használsz, így értelmezni tudod az

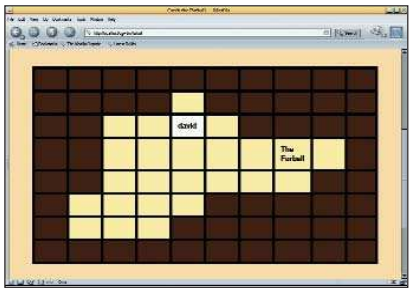


adatokat és módosíthatsz rajtuk a böngésző segítségével. Előfeltételei: libgtk, libgdk, libgmodule, libglib, libdl, libXi, libXext, libX11, libm, libnetsnmp, libwrap, glibc, libcrypto, libnsl.

➲ http://www.kill-9.org/mbrowse/index.html

Furball

Szórakoztató játék gyerekeknek, de ha átszerkeszted, bárki számára élvezhetővé teheted. A játék során a többieket



kérdések és más eszközök segítségével ismerheted meg. Afféle „ki a gyilkos?” játék. Futtatásához webkiszolgáló, webböngésző és Python szükséges.

➲ http://www.claws-and-paws.com/software

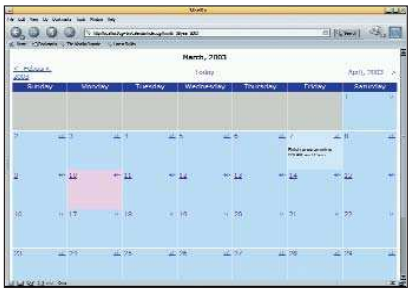
AeroMail

Aki spártaian egyszerű webes levelezőügynökre vágyik, válassza az AeroMailt. Mentes minden sallangtól, még címjegyzék sincs benne, viszont lehetővé teszi a levélkiszolgálónak IMAP-protokollon keresztül történő elérését. A levelek tárolására és csoportosítására mappákat használ. Az üzenetekre válaszolhatsz, továbbíthatod őket, illetve új üzenetet szerkeszthetsz. Minden lényeges szolgáltatást tartalmaz, de ezeken kívül nem sok egyebet. Kiválóan alkalmas utazók számára, akik visszatéréskor valamelyik szokásos levelezőügyfelet használják. Futtatásához webkiszolgáló, IMAP-támogatással fordított PHP és IMAP-kiszolgáló szükséges.

➲ http://the.cushman.net/projects/aeromail/index.php

My Calendar

Ha csak egyetlen naptárra van szükséged (nem mindenkinek, csak neked, az irodának vagy a világhálónak), ez a program rendkívül egyszerűen és gyorsan telepíthető. Mindaddig, amíg a *protected*/ könyvtár védett, nem kell aggódnod amiatt, hogy valaki esetleg megváltoztatja a bejegyzéseket. Ezenkívül a naptár elektronikus levélben értesít a következő



három napra beütemezett megbeszélésekről, amennyiben beállítod a `cron`-t, hogy futtassa a levelező parancsfájlt. Futtatásához webkiszolgáló, Perl, cal és (nem kötelezően) pscal szükséges.

➲ http://fuzzymonkey.org/newfuzzy/software

ps-watcher

Ha szükséged van egy olyan eszközre, ami szemmel tartja a folyamattáblát és annak alapján elvégez bizonyos műveleteket, a ps-watcher programot ajánlom. Nap mint nap kóbor Netscape-folyamatokra kellett vadásznom, és ki kellett irtanom őket. A ps-watcher remekül ellátja ezt a feladatot, még mielőtt a rendszer feldobná a talpát. A felhasznált processzoridő, illetve más jellemzők szerint ütemezheted a tevékenységeket. Egyszerűen határozd meg a jellemzőt a beállításfájlban, és állítsd be az elvégzendő műveletet, ami lehet `log`, `kill`, `log` és `kill`, de sok egyéb is. Futtatásához Perl és az alábbi Perl-modulok szükségesek: Sys::Syslog, File::Basename, Pod::Text, Config::IniFiles, Getopt::Long.

➲ http://ps-watcher.sourceforge.net

Linux Journal 2003. június, 109. szám

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