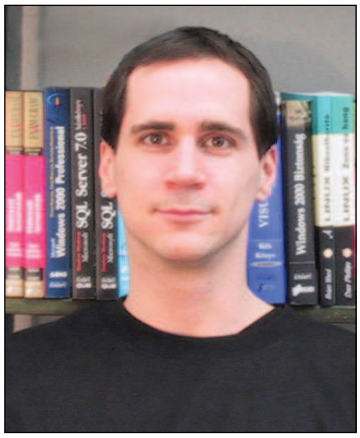
Beköszöntő



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

Végre megjelent a harmincnegyedik számunk, ami azt is jelenti, hogy háromévesek lettünk. Ezt a matematikailag nehézkesen bizonyítható kijelentést úgy is megfogalmazhattam volna, hogy három évvel ezelőtt, novemberben jelent meg az ország méltán legnépszerűbb Linuxszal foglalkozó nyomtatott sajtóorgánumának első száma. De ez talán egy csöppet fellengzősen hangzana.

Az első szám még irkafűzéssel, sok lelkesedéssel és még több sajtóhibával látott napvilágot. Azóta sokat változtunk, például nem irkafűzéssel jelenünk meg. A másik kettő még megvan. Ugyanúgy, ahogy a parázs viták is, amelyeket a fordításokkal kapcsán folytatunk, vagy a siránkozások, hogy nyugaton mennyivel szélesebb körben használják a GNU/Linuxot, mint idehaza. Bár a hangsúlyok eltolódtak.

Ma már nem azon kapunk hajba, hogy le kell-e fordítani a server szót, csupán diplomatikusan dörmögünk, hogy de jó lenne a backportra a visszaültetésnél jobb fordítást találni. És ha körbenézünk láthatjuk, ahol az emberek Linuxot akarnak használni, ott már szabad utat kapnak. Oktatásban, munkahelyen, otthon.

Na nem mintha mindenhol sajttal lenne kikövezve az út a GNU/Linux előtt. A minap például egy laptoppal vesződtem (márkás, drága, új, csilli-villi), amihez adtak Windows XP-t. Pontosabban előtelepítették nekem a teljes lemezre. Ha akartam újratelepíthettem, s egy csöpp szöveg figyelmeztetett, hogy a telepítő egyedi, abban az értelemben, hogy első lépésként fűbe vágja az egész merevlemezt, ledarálja az összes adatot, és feltesz nekem egy gyárilag előtelepített Windowst. Hiába na, angyalok ezek az OEM-rendszerek. Azért (így utolsóként) a „pénzes” területek is kezdenek alkalmazkodni. Már vannak helyek, ahol még azt a plusz két-háromtízezrest sem kell kifizetnünk, ha esetleg más operációs rendszert akarunk a gépünkhöz. Idő kérdése, és talán eljutunk a nyugaton (és keleten) immár több éve szokásos módihoz: megkérdezik a kereskedők, mit szeretnénk. Mert ugye az elég fontos. Legalábbis nekünk. A vevő pedig olyan, hogy oda megy, ahol azt kapja, ami kell neki, nem azt, amiről úgy tudja az eladó, hogy kell neki. Jó esetben…

Elsődleges feladatunk tehát ma már nem az, hogy kiharcoljuk a Linux számára az életteret, hanem az, hogy megmutassuk az embereknek, hogy létezik, él és virul ez a rendszer, a kedves felhasználó választhatja ezt is, ilyen és ilyen előnyeivel, ilyen és ilyen hátrányaival együtt. Remélem, hogy ezen a területen tudunk segítséget nyújtani új olvasóink számára, amellett, hogy a gyakorló rendszergazdák, programozók továbbra is megtalálják a számukra érdekes cikkeket, írásokat.

Kívánok mindenkinek boldog linuxozást!

Programvadászat

Mindi-Mondo

Ez a programpáros igen jól használható összeomlás utáni helyreállítónak. Segítségükkel a rendszer pillanatnyi állapotáról biztonsági mentést készíthetünk, amit egy esetleges rendszerösszeomlás után csak beteszünk a CD-olvasóba, elindítjuk a gépet, és szinte önműködően visszakapjuk a régi rendszerünket. A magazin 22. oldalán található cikkben a program használatáról bővebben is olvashatunk.
Nézzük meg, miként is telepíthetjük fel ezt a hasznos programpárost. Fűzzük be a CD-t!

```
# mount /mnt/cdrom
# cd /mnt/cdrom/Magazin/
➥Mondo-Mindi
# ./setup
# cd /
# umount /mnt/cdrom
```

Ez a módszer remekül működik az alábbi rendszereken:

- Red Hat 7.x, 8.0, 9
- Mandrake Linux 8.x, 9.x
- SME/e-Smith 5.6
- SuSE Linux 8.2
- Trustix 2.0
- TurboLinux Server 6.5
- TurboLinux Workstation 7.0

Nem lett kipróbálva, de ettől még van esély a működésre a következő változatoknál:

- Caldera OpenLinux Workstation 3.1
- Caldera OpenLinux Server 3.1
- Mandrake Linux 8.0

Az alábbi terjesztésekkel viszont biztosan nem működik:

- Slackware 8.0
- Debian 3.0
- TurboLinux Server 6.0.5

A Debian telepítését egy `apt-get install mondo` paranccsal könnyedén megoldhatjuk.
Mint az az *első képen* is jól látható, szinte mindenféle médiára készíthetünk mentést, nem állhat utunkba sem a CD, CD-RW, merevlemez, sem az NFS kötetek. Kényelmes és gyorsan megszokható, megszerethető programokról van szó, használatával nem kell félnünk egy esetleges rendszerösszeomlás következményeitől – amennyiben elég friss a mentésünk.

OpenOffice.org 1.1

Megjelent a linuxos körökben legnépszerűbb (egyben legnagyobb és leglassabb) irodai csomag következő megbízható változata, az 1.1-es. Az előző változatokhoz képest érezhető sebességnövekedést tapasztalhatunk, ami ennek a programcsomagnak nagymértékben az előnyére válik.
Telepítése a szokásos módon zajlik, azaz kezdjük a fájlok kicsomagolásával:

```
#mkdir install
#cd install
#mount /cdrom
#tar -xvzf
/cdrom/OpenOffice.org
```

ekkor létrejön a ***Ooo_1.1.0_LinuxIntel_install*** könyvtár

```
#cd Ooo_1.1.0_LinuxIntel_
➥install
#./setup
```

Amennyiben többfelhasználós üzemmódban szeretnénk telepíteni, akkor a `#./setup -net` parancsot használjuk rendszergazdaként. Így ha több felhasználó is dolgozik a gépen, akkor nem kell mindenkinek az egész csomagot feltelepítenie, ugyanis ez nagy erőforráspazarlás lenne, hanem elég csak egyszer felpakolni. Ezt követően csupán a felhasználók fájljait kell másolni.

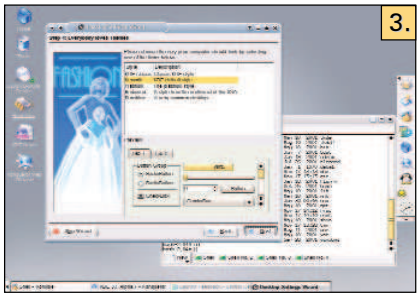
SELinux

33. oldalon kezdődő cikkünk témája az NSA (National Security Agency) által fejlesztett SE Linux (Security Enhanced Linux). Hogy mindenki kipróbálhassa

a cikkben leírtakat, lemezmellékletünk SELinux könyvtárában megtalálható minden csomag, ami ehhez szükséges.

KDE 3.1-folytatás

A mostani korongon folytatjuk a KDE-felület közreadását, most a Red Hat Linux 9-es változatához adjuk közre a csomagokat.



Rendszermag

Gyors ütemben fejlődik a 2.6-os rendszermag, már a 2.6.0-test7-es változatnál tart, egyre közelebb kerülve a teljes üzembiztossághoz. Valószínűleg ennek a rendszermagnak kell majd beteljesítenie a 2.4-es sorozathoz fűzött nagy reményeket, hogy a Linux nagyvállalati környezetben is igazán elismert legyen.

Játék

A játékrovathoz kapcsolódó anyagok a szokásos helyen a Jatek könyvtárban találhatóak. Képeket és filmet egyaránt találhatunk a hamarosan megjelenő hazai fejlesztésű T3 játékról.

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

Palm-újdonságok

Az elmúlt időszakban három új tenyérgépet is bemutatott a Palm. Az alsó kategóriás Zire 21 nem éppen az élvonalba tartozik, ám kedvező, 99 dolláros árával, 8 MB memóriájával és szürkeárnyalatos

kijelzőjével alkalmas arra, hogy a cég minden érdeklődő számára valamilyen eszközt tudjon ajánlani. A Tungsten E a felsőbb kategória egy feljavított darabja, 32 MB memóriával, frissített alkalmazásokkal és 199 dolláros árral. Igazi újdonságként a Tungsten T3 könyvelhető el, amelynek kijelzője nemcsak hagyományos álló, de fekvő módban is használható, és megnövelt, 320×480 képpontos felbontással bír. Processzora 400 MHz-es órajellel üzemel, 64 MB memóriával látták el, így irodai dokumentumok kezelésére, mozgóképek lejátszására is alkalmas. Ára ennek megfelelően kevésbé barátságos, 399 dollár. Ugyancsak a Palmhoz kötődik, hogy a cég tenyérgépeire is elérhetővé vált az IBM WebSphere Micro Environment Java virtuális gépe. Akik 2003. október 1-je után vásároltak Tungsten-gépet, azok ingyenesen tölthetik le a telepítőkészletet, a többi felhasználónak 6 dollárt kell fizetnie érte. A két cég elsősorban a nagyvállalati felhasználókat célozza meg ajánlatával, akik a Java szélesebb körű elérhetőségével szorosabban egybeépített megoldásokat hozhatnak létre.
➲ http://www.palm.com

Linuxos Homeworld

A Relic Entertainment – miközben Windows alá már a második változatot jelentette meg – elérhetővé tette a Homeworld játék forráskódját. A kód nem GPL szerződéssel jelent meg, és a hangrendszer fejlesztői készletét sem tartalmazza, minden mást azonban igen, így legalábbis tanulási célokra érdemes lehet beszerezni. Lelkes rajongók dolgoznak a Linuxra átültetett változaton is, amelynek előzetes kiadása forráskódként és bináris formátumban egyaránt letölthető.
➲ http://www.relic.com/rdn

Samsung a Napsterhez

A Samsung bejelentette, hogy olyan új digitális zenelejátszó készülékeket fog piacra dobni, amelyek szorosan kötődnek majd a hamarosan induló Napster 2.0 szolgáltatáshoz. A cég világszerte a fiatalok meghódítását tűzte ki célul, és többek között beépített tévével rendelkező mobiltelefonokkal, hatalmas plazmatévékkel készül a digitális korszak ifjú híveinek zsebében lapuló pénztárca megcsapolására. A zeneszolgáltatáshoz a mintát az Apple iPod-lejátszókkal társuló iTunes szolgáltatása adja, amelynek példátlan sikere több cég számára is egy lórúgással ért fel: valamiről megint lemaradtak. Az eredeti Napster annak idején még törvénytelen fájlcserélő hálózatként működött, amíg egy bírósági döntéssel fel nem számolták. Vagyonát a Roxio szerezte meg, a cég a néhai Napster alapjaira építkezve egy immár jogtiszta szolgáltatás az év őszi indításán munkálkodik.
➲ http://www.samsung.com

Előtelepített LindowsOS gyártóknak

A Seagate és a Lindows.com együttműködésének eredményeként született meg a LindowsHD. Az új termék valójában egy teljesen átlagos, asztali gépekbe szánt, 40 GB-os Seagate Barracuda merevlemez, amelyre előtelepített LindowsOS 4.0 kerül. A meghajtót decembertől elsősorban (amerikai) számítógépgyártóknak kínálják, akik a teljesen üres lemezekhez képest felár nélkül vásárolhatják meg őket. Így nem kell külön foglalkozniuk az operációs rendszer telepítésével, beállításával, szerelés után azonnal kész gépet adhatnak át a vásárlóknak. A „nem márkás" gyártók a tengerentúlon is a gépeladások több mint felét mondhatják magukénak, most a LindowsHD használatával minden egyes számítógépen 100 dollárt takaríthatnak meg – az összeg kísértetiesen emlékeztet az OEM Windows XP Home árára, talán nem is véletlenül. Ide kapcsolódik, hogy a Lindows.com LindowsEspanol névvel bejelentette operációs rendszerének spanyol nyelvű változatát. A spanyol nyelvet világszerte 332 millió ember beszéli, ám a cég vezetőit valószínűleg jobban érdeklik azok a milliók, akik az Egyesült Államok határain belül, illetve Közép- és Dél-Amerikában élnek, és olcsó számítógépet szeretnének vásárolni. Lehet, hogy nem ez a réteg a legtehetősebb, de a Lindows.com hosszú távú terveihez jelentős tömegbázist adhat.
➲ http://www.lindows.com/lindowsHD

Vigyáz magára

Az IBM megújult ThinkPad termékcsaládjában két újfajta adatbiztonsági megoldással felruházott típust – R50 és T41 – is találunk. Mivel a hordozható számítógépeket tulajdonosaik sokszor leejtik vagy nekiütik valaminek – a kimutatások szerint a vállalatoknál minden évben a hordozható gépek egytizedét használóik

összetörik –, az IBM egy külön lapkával szerelte fel gépeit, amely erős gyorsulást érzékelve képes a gép merevlemezében található író–olvasó fejet parkolópályára helyezni. Így nem kell attól tartani, hogy az ütés hatására elmozduló olvasófej a merevlemez korongjának ténylegesen adatokat tároló részét megkarcolja, és emiatt adatvesztés következik be. Igaz, hogy maga a gép ettől nem menekül meg, ám a rajta tárolt adatok fennmaradására jó esély mutatkozik.
➲ http://www.ibm.com/thinkpad

Eldobható CD-k

A Sanyo optikai adathordozókat fejlesztő Sanyo Mavic Media nevű leányvállalata bejelentette, hogy hamarosan megkezdi növényi alapanyagokból készült CD-inek kereskedelmi forgalmazását. A MildDisc lemezek különleges, polimerizált tejsavból készülnek, amelynek az alapanyaga egyszerű kukorica. Megfelelő kémiai eljárásokkal ezt az anyagot úgy alakítják, hogy a műanyagokhoz hasonló tulajdonságokat vegyen fel, és alkalmas legyen CD-k kialakítására. Egyetlen lemez legyártásához mindössze 85 kukoricaszemre van szükség, vagyis egyetlen csőből elméletileg akár tíz lemez alapanyaga is előállítható. A kukoricalemezek nemcsak elállításuk, de utóéletük során is kímélik a környezetet, ugyanis a természetben előforduló mikroorganizmusok kiváló étvágyának köszönhetően nyomtalanul, káros anyag hátrahagyása nélkül bomlanak le. Ha nemcsak magát a lemezt, hanem a tokot és a csomagoló fóliát is ilyen műanyagból készítik, akkor – a jelenlegi CD-kkel ellentétben – a teljes termék környezetbarát módon tüntethető el, amikor már nincs szükségünk rá.

Bemutatkozik az új nemzedék

A Pannon GSM és a Nokia együttműködésével hazánkban is működésbe lépett az első harmadik generációs (UMTS) mobil távközlési rendszer – egyelőre bemutató jelleggel. Az új hálózatok a jelenleginél gyorsabb adattovábbítást tesznek majd lehetővé, megnyitva az utat a gyors letöltések, a videokonferenciák és a különféle multimédiás szolgáltatások előtt. Nyugat-Európában a 3G hálózatok kiépítésének jogát a távközlési cégek hatalmas összegekért vásárolták meg, ám a felhasználók száma rendre elmaradt a várttól. Hazákban – mivel még a tendert sem írták ki – valószínűleg megfontoltabban fogják végrehajtani a fejlesztéseket, ugyanakkor a már jelen lévő mobilcégek megkezdték a felkészülést a majdani szolgáltatások beindításához.

A brazilok is a Linux mellett döntöttek

A brazil kormány és az IBM közös szándéknyilatkozatában foglaltak szerint a két fél együtt fog azon fáradozni, hogy segítse a nyílt szabványokra és programokra alapuló megoldások, köztük a Linux elterjedését a dél-amerikai országban. Első lépésben a hátteret kívánják megteremteni ahhoz, hogy a nyilvános szektorban – ami nem tiszta, hogy mit takar – elterjedjen a Linux, és általa alacsonyabb költséggel lehessen egymással együttműködni képes rendszereket építeni. Az IBM nemrég az orosz kormánnyal is megegyezett egy linuxos központ létrehozásáról. Az eseményeket szemlélve egyrészt arra következtethetünk, hogy az IBM linuxos részlegénél év végén elég szép jutalmakat fognak osztani, másrészt pedig arra, hogy a fejlődőnek nevezett országok is rádöbbentek: a nyílt operációs rendszer térhódítását ők sem hagyhatják figyelmen kívül. A kérdés már csak az, hogy hozzá tudnak-e adni valamit a tortához, vagy csak egy szeletkét kérnek belőle.

Biztonsági szabvány asztali operációs rendszerekhez

P2200 jelzéssel új szabvány kidolgozásában vágott bele az IEEE. A Base Operating System Security, rövidítve BOSS az általános célú, kereskedelemben kapható, a polcról készen leemelhető operációs rendszerekre vonatkozóan fog biztonsági elvárásokat megfogalmazni. Hiába ugyanis a megannyi biztonsági intézkedés és felügyeleti eljárás a vállalatoknál, ha az operációs rendszerek nem nyújtanak kellő segítséget a biztonságos környezet megteremtéséhez. A BOSS a belső hibákra és külső fenyegetésekre egyaránt kitér majd, a hiányosságokat gyökerüknél, a tervezési szakaszban próbálja megragadni. Összeállítói minden biztonsággal foglalkozó szakembert az előírás-gyűjtemény összeállításában való részvételre buzdítanak, ám végeredményként nemcsak a műszaki, de a gazdasági vagy irányító szerepet betöltő szakemberek számára is áttekinthető, jól értelmezhető dokumentumot szeretnének létrehozni. A szabványt – amely alapvetően az ISO Common Criteria rendszerének gondolatát követi – az IEEE gyorsított ütemben készíti, megjelenése 2004 végére várható.

➲ http://www.bosswg.org

Ne rád várjak…

A valós idejű rendszereket fejlesztő FSMLabs olyan Linux-változatot készített, amely kevesebb mint 200 ms alatt képes betölteni magát. A jelenlegi változatok indítása akár öt másodpercnél is tovább tarthat, ami egy asztali számítógépnél ugyan nem lenne sok, ám egy beágyazott operációs rendszert futtató készüléknél elfogadhatatlan – az ilyen gépektől használóik azt várják, hogy bekapcsolás után gyakorlatilag azonnal rendelkezésükre álljanak. Az FSMLabs fejlesztői tanulmányozták az operációs rendszer indításának a folyamatát, meghatározták azokat a pontokat, amelyeken késleltetés lép fel, majd a megfelelő eljárások módosításával csökkentették a várakozások időtartamát. A cég által alkalmazott módszerek processzortípustól függetlenül alkalmazhatók. További finomításuk révén várhatóan 0,1 másodperc alatti rendszerindítási időket is el lehet majd érni, illetve megoldásaik az x86 alapú rendszerekben – vagyis az asztali számítógépeken – is megjelenhetnek.

➲ http://www.fsmlabs.com

Linuxos cégadatbázis

Az UHU-Linux csapat elindította a linuxos cégeket összegyűjtő weboldalát. Célkitűzésük az, hogy segítsék közelebb hozni egymáshoz a Linux iránt érdeklődő felhasználókat és a (bármilyen) Linux-terjesztéssel foglalkozó vállalkozásokat.

➲ http://www.uhulinux.hu/cegek

Educatio 2003

Idén is megrendezik november 27. és 30. között, a Budapesti Vásárközpont A pavilonjában a magyar oktatás legjelentősebb és legrangosabb hagyományos szakmai seregszemléjét. Több száz kiállító – köztük valamennyi magyar egyetem és főiskola – képviseli a hazai közép- és felsőoktatást, a taneszközgyártást, a tankönyvkiadást, a szakképzést, a nyelvoktatást, a minőségbiztosítást és az informatikát. A látogatók a kiállítási standok mellett szakmai előadásokon és bemutatókon ismerkedhetnek az oktatás jelenével és jövőjével.
A rendezők az előző évi kiállítás alapján 5-600 kiállítóra, és a négy nap során 60–70 ezer látogatóra – középiskolás diákokra, szüleikre, pedagógusokra és intézményvezetőkre – számítanak.
A kiállítás minden belépő számára ingyenes és minden nap 10–17 óráig tart nyitva.

➲ http://www.sulinet.hu/tart/cikk/nd/0/16191/1

Az EU a szabadalmak ellen

Az Európa Parlament a számítástechnika jövőjét illetően fontos döntést hozott szeptember 24-én: az Európai Unióban a jövőben alapvetően nem engedik a programokban megvalósított megoldások levédését. Az eredeti tervezet alapján a programgyártók viszonylag szabad kezet kaptak volna a különféle szabadalmak bejegyzéséhez, ezzel gátolva, hogy mások is hasonló megoldásokat készítsenek. A tervezet nagy felháborodást váltott ki, hiszen a kisebb cégeknek nincs elég erőforrásuk ahhoz, hogy szabadalmaikat kivétel nélkül bejegyeztessék, az ingyenes programok készítői pedig legfeljebb zsebből fedezhették volna a költségeket. Az eredetileg beterjesztett javaslatot nem kevesebb, mint kilencven módosító indítvánnyal változtatták meg, ennek eredményeként a szigorú amerikainál sokkal lazább szabályozás jött létre, amely maguknak a programtermékeknek a szabadalmazását nem teszi lehetővé.

➲ http://www.europarl.eu.int

Okosodó vezeték nélküli hálózatok

Az Atheros bemutatta negyedik generációs WLAN-lapkáinak első példányait. Az új, a korábbiaknál érzékenyebb lapkák révén megkétszerezhető a vezeték nélküli hálózatokkal áthidalható távolság, és a korábbi eszközökhöz képest mintegy hatvan százalékkal csökkenthető az energiafogyasztás is. Sokkal érdekesebb azonban a lapkák két különleges szolgáltatása, a Wake-on-Wireless és a Wake-on-Theft. Az előbbi révén az elsősorban vállalati környezetekben használatos felügyeleti megoldások terjeszthetők ki a vezeték nélküli eszközökre is, míg az utóbbi lehetővé teszi a hordozható eszközöknek a vállalati hálózatból történő engedély nélküli eltávolításának az észlelését, akár az adott mobilgép kikapcsolt állapota mellett is. Az Atheros új eszközei – legalábbis új szolgáltatásaikat tekintve – természetesen nem képesek együttműködni a korábbi készülékekkel, így használatukhoz a hozzáférési pontok cseréjére is szükség lesz. A cég lapkáira épülő termékek várhatóan hamarosan több nagy gyártó – D-Link, Linksys, SMC és mások – kínálatában is megjelennek.

➲ http://www.atheros.com

Megújuló PCMCIA

A PCMCIA kereskedelmi szövetség hivatalosan is megjelentette a főleg mobil eszközökben használatos PCMCIA, pontosabban immár ExpressCard szabvány legújabb változatát. Az új szabvány alapján vékonyabb, könnyebb, gyorsabb kártyák illeszthetők a mobilgépekhez. Az új kártyák valójában csak formájukban jelentenek újdonságot, illesztőfelületükben kevésbé.

A szövetség tagjai úgy határoztak, nem alkotnak teljesen új illesztést, inkább két nagysebességű buszt használnak fel: a PCI Expresst és az USB 2.0-t. Mivel a legújabb noteszgépek már most is mindkettőt támogatják, az ExpressCard kártyákra való áttérés viszonylag fájdalommentesen végrehajtható. ExpressCard kártyát kétféle változatban vásárolhatunk majd. A kártyák vastagsága mindkét esetben 5 mm, hosszúsága 75 mm, illesztőcsatlakozása 34 mm lesz, ám az egyik változat szerinti kártyák 54 mm-re szélesednek ki – az öblösödésre apró merevlemezes meghajtók korongjainak elhelyezése, illetve a kártyaolvasó eszközök helyigénye miatt van szükség.

A PCMCIA elnevezéssel (Personal Computer Memory Card Industry Association) eredetileg kisméretű memóriakártyákat illettek. A szabványt hamarosan módosították, így modemeket, hálózati kártyákat és egyéb, nem memória jellegű eszközöket is lehetett ilyen felületen keresztül a gépekhez illeszteni. A név hamarosan PC-kártyára változott, hangsúlyozva, hogy ezek az eszközök többé nem csak adattárolásra használhatók. Akkoriban a kártyák még a jó öreg ISA buszt használták, majd amikor ezt a PCI busz váltotta fel, az elnevezés ismét módosult: ettől kezdve CardBus kártyákat vásárolhattunk.

A legújabb nemzedék, az ExpressCard tehát csak késői leszármazottja a PCMCIA-nak, ám az elnevezés bizonyára sokáig élni fog a köztudatban.

➲ http://www.pcmcia.org

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

Az SCO-s botrány kronológiája

A Linuxvilág októberi számának 70. oldalán szerkesztői figyelmetlenségből egy cikket közöltünk SCO Linux 4 címmel. Ennek orvoslására a mostani lapszámban az SCO pereskedésének hátterét és állomásait vesszük sorra.

2002. augusztus

A Caldera nevet változtat

A Caldera International Inc. bejelentette, hogy a részvényesek kívánságának megfelelően a nevét The SCO Group (SCO)-ra változtatja. A Caldera OpenLinux következő változata már az SCO Linux 4.0 nevet fogja viselni. A bejáratott márkanévhez, a két évvel ezelőtt megvásárolt SCO-hoz nyúl (SCO OpenServer és SCO UnixWare), amely már több mint két évtizede van a piacon.

2003. március

Feszültség a UnitedLinux-csoporton belül

Az SCO Group, amely linuxos üzletén kívül az eredeti AT&T UNIX-hoz fűződő jogok tulajdonosa és az SCO UNIX márkát továbbvivő cég is, keresetet adott be az IBM ellen. Az SCO egymilliárd dollárt követel az óriáscégtől, mert szerinte az IBM megsértette a Unix technológia felhasználási szerződésének feltételeit, és lehetővé tette e módszerek Linuxba való átszivárgását.

2003. április

Az SCO Group az IBM után a nagyobb Linux-terjesztőkkel is el szeretne számolni

Egy áprilisban közzétett interjúban az SCO Group (korábban Caldera International) elnök–vezérigazgatója, *Darl McBride* elmondta, hogy amennyiben a bíróság előtt is sikerül érvényt szerezniük az IBM-mel szembeni követeléseiknek, benyújtják a számlát a kereskedelmi Linux-változatok piacát csaknem teljesen lefedő két cégnek: a Red Hatnek, illetve a SuSE-nek is.

2003. május

- *A SCO szerint a Linux-rendszermag tiszta, de a SuSE és a Red Hat „bűnös”*
  A SCO-Caldera munkatársa, *Chris Sontag* érdekes nyilatkozatot adott a MozillaQuest.com-nak, amelyben kifejtette, hogy a SCO a Linux-rendszermagban nem talált SCO-tulajdonú kódot, de a német SuSE és a piros kalapos Red Hat jobb, ha elkezd ügyvédre gyűjteni.
- *SCO: Unix-kód került a Linuxba*
- *UnixWare-kódrészek a Linux-rendszermagban?*
  Darl McBride szerint Unix-kódsorok kerültek a Linux szívébe, magába a rendszermagba. Ezek a kódok részben módosítás nélkül, részben módosításokkal álcázva szerepelnek a magban.
  McBride elutasított minden olyan kérdést, amely a részletekre utalt. Az állítólagosan beépített kódrészekről csak ennyit mondott: „némelyikük néhány évvel ezelőtti kódrész, de vannak újak is és – tette hozzá – a másolás mértéke nem kicsi.”
- *Internetes támadás bénította meg a Linuxot kódlopással vádoló SCO weboldalát.*
- *Törvénytelen lehet a Linux használata*
  Az SCO Group bejelentésében figyelmeztette a Linux-felhasználókat, hogy jogi kérdéseket vethet fel, amennyiben továbbra is folytatják a nyílt forrású operációs rendszer használatát. Az SCO szerint ugyanis a Linux számos összetevője a cég által szabadalmaztatott eljárást valósít meg, vagy egyenesen az ő Unix-változatukból származó kódot hasznosít engedély nélkül.
- *Linus Torvalds sejti, hogy miért függesztette fel Linux-eladásait az SCO Group*
  Az ok valószínűleg az lehet, hogy amíg ők maguk is terjesztik a Linuxot, addig rájuk is vonatkoznak a Linux-rendszereket felépítő programok legtöbbjéhez csatolt GNU GPL szerződés feltételei, a GPL-ben pedig egyebek közt az is szerepel, hogy a „vevők” jogait a program kibocsátója nem korlátozhatja.
- *A UnitedLinux az SCO Group nélkül folytatja tovább?*
  Az SCO megszüntette a UnitedLinux csoporttal való marketinges, üzletfejlesztési és pénzügyi együttműködését, és nem biztosít további erőforrásokat a konzorcium számára, így kétségessé vált a cég további tagsága.
- *Online felhívásban támadják az SCO-t*
- *A Microsoft megvásárolta a Unix felhasználási jogát*
  A Microsoft bejelentette, hogy licenceli az SCO Grouptól a Unix operációs rendszerrel kapcsolatos technológiákat, szabadalmakat, valamint hozzáférést kap a forráskódhoz is.
- *A Sun védett, a HP és a Red Hat nem*
  Újabb, a Linuxokban érdekelt vállalatok adtak hangot az ügyben, és úgy tűnik, a nagyok egy része az SCO mellé áll (de legalábbis nem az IBM-hez csatlakozik).

2003. június

Jogtalan az SCO Linux ellenes eljárása?
Újabb gigász a harcban…

A Novell levelet küldött az SCO számára – ebben szereplő pontos kérdéseire pontos választ vár. A szintén Unix-jogokkal rendelkező cég ugyanis arra szólította fel „kollégáját”, hogy nevezze meg, mely Unix-forráskódok kerültek át a Linuxba.

- *Betyárokat toboroznak az SCO Group ellen*
  Raymond olyan Unix-hackerek jelentkezését várja, akik munkájuk során már hozzáfértek valamelyik jogvédett Unix binárisain és dokumentációján kívül a rendszer forráskódjához úgy, hogy nem kellett aláírniuk harmadik felet kizáró titoktartási szerződést (non-disclosure agreement, NDA).
- *Az SCO-t kiengesztelné, ha az IBM felvásárolná a céget…*
  Az SCO elnök-vezérigazgatója a múlt héten megerősítette, hogy amennyiben példának okáért az IBM felvásárolná a céget, rövid úton véget érhetne a Linux ellen indított hadművelet.
- *A kényes rendszereknél ne használjunk Linuxot?!*
  A neves Gartner elemzőcég azt javasolja, hogy a

fontos rendszerek esetében a cégek minimalizálják a Linux használatát az SCO Group indította jogi felelősséget firtató eljárás bizonytalan kimenete miatt.
- *Az SCO Group Linux-ellenes kampánya saját weboldalának átmeneti bezárásához vezetett*
- *Péntek 13: az IBM péntek éjfélig kapott haladékot*
- *Linux-forráskód az SCO Group Unix-változatában?*
  Nyílt forráskódú programokkal foglalkozó fejlesztők egy csoportja szerint előfordulhat, hogy az SCO Group megsértette a Linux-rendszerek számos alkotóelemére vonatkozó GNU GPL felhasználási szerződés feltételeit azzal, hogy Linux-forráskódot használt fel az SCO Unix Linux Kernel Personality (LKP) rétegében, és a kódmódosításait nem hozta nyilvánosságra, illetve nem tüntette fel a megfelelő helyen a vonatkozó szerzői jogi adatokat.
- *Elveszítheti Unix-felhasználási szerződését az IBM*
- *A jogászok szerint az SCO az esélyesebb…*
- *Perrel fenyegette meg az SCO-t egy Linux-rendszermag-fejlesztő*
- *SCO: elsőként a vezetőség száll ki?*
- Egy névtelen felhasználó által beküldött híradás szerint (➲ http://newsvac.newsforge.com/article.pl?sid=03/06/09/139257) az SCO-csoport egyik alelnöke a hónap elején megszabadult az összes általa birtokolt SCO-részvénytől (15 000 darabtól, egyenként 6 dollár értékben). Annak fényében, hogy az SCO mindenhol azt híreszteli, hogy meg fogja nyerni az IBM ellen indított egymilliárd dolláros kártérítési pert, ez igencsak elgondolkodtató.
- *Az SCO szerint törvénytelen az IBM AIX további használata*
- *Linust is perli az SCO*
- *Már hárommilliárd dollár a jogtalanul „felturbózott" Linuxok ára*
- Az SCO szerint az IBM úgy tette a Linuxok számára elérhetővé a csoporttól licencelt Unix-technológiákat, hogy nem fizette meg azok után a jogdíjakat. Márciusi egymilliárd dolláros kárigényét az SCO hárommilliárdra növelte.
- *Linux-licencprogramon dolgozik az SCO*

2003. július

Gates: Microsofttól „kölcsönzött" elemeket is tartalmaz a Linux

2003. augusztus

- *A Red Hat perli az SCO-t*
  A vezető Linux-cég keresetet adott be az SCO Group ellen, mert a bírósággal akarja kimondatni, hogy a technológiái nem sértenek SCO szellemi tulajdont, ugyanakkor kártérítést követel az SCO „tisztességtelen és megtévesztő" gyakorlata miatt.
- *Az SCO megjelentette felhasználási szerződésének feltételeit*
  Egy munkaállomás felhasználási szerződésének költsége 199 dollár.
  A kiszolgáló felhasználási szerződésének költsége a processzorszámtól függően alakul az alábbiak szerint:
  1 processzor 699 dollár
  2 processzor 1149 dollár
  4 processzor 2499 dollár
  8 processzor 4999 dollár
  Minden további processzor 749 dollár.
- *Az IBM viszontkereseteket nyújtott be az SCO Group ellen*
- *A Red Hat szerint az SCO fel fogja adni*
  Minden valószínűség szerint idő előtt vissza fog lépni az IBM ellen benyújtott, mostanra már hárommilliárdosra duzzadt keresetétől az SCO – állítja *Mark Webbink*, a Red Hat elnökhelyettese. Webbink szerint az SCO stratégiája arra utal, hogy a cég nem fogja megvárni, amíg bírósági döntés születik az ügyben.

2003. szeptember

- *A Red Hat elindította az Open Source Now alapítványt*
  Augusztus elején a Red Hat azt ígérte, hogy egy alapítványt hoz létre, amelynek a célja az lesz, hogy megvédje az nyílt forrású programfejlesztéseket és a Nyílt Forrás Közösségét a SCO támadásától.
- *A hatvanas évekből származó kódrészletekkel próbálja hitelesíteni vádjait az SCO Group.*
- *Fizetnie kell a SCO-nak*
  Megszületett az első bírósági döntés, amelynek értelmében a SCO-t arra kötelezte a bíróság, hogy kártérítést fizessen valótlan állításai miatt, miszerint a Linuxban lopott System V-kód van.
- *Az IBM új vádja: az SCO szerzői jogot sértett*
  A kék óriás kiszélesítette a frontot az SCO ellen indított ellentámadásában, miután csütörtökön beadott újabb keresetében már az IBM szerzői jogainak elbitorlásával vádolta meg a céget.

2003. október

- *A SCO már az SGI-t támadja*
  Egyes hírek szerint az SGI figyelmeztetést kapott a SCO-tól, amelyben az áll, hogy a SCO megvonni készül a Silicon Graphicstól a Unix System V felhasználási jogát. Hogy mi alapon teszi ezt? Szerinte az SGI megsértette a két cég közötti szerződést.

Kíváncsian várjuk, kik lesznek a következők.

Források

➲ http://www.index.hu
➲ http://www.terminal.hu
➲ http://www.hwsw.hu
➲ http://www.hup.hu
➲ http://www.linux.hu
➲ http://www.prog.hu

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

GNU/Linux Szakmai Konferencia

Egyesületünk már ötödik alkalommal rendezi meg szokásos éves szakmai rendezvényét. A helyszín ismét Budapest, választásunk most a Hotel Novotel Budapest Centrumra esett (Budapest, VIII., Rákóczi út 43–45.).
Időpont: 2003. november 08. 10–18 óráig
Regisztráció: reggel 9 órától
Terveink szerint 400 támogatott és 100 üzleti belépős látogatóval számolhatunk. Az üzleti látogatók létszáma szükség esetén kis mértékben emelhető. A rendezvény fő támogatója a KonzumBank Rt., képviselői részletesen beszámolnak majd a Linux bankjukban történő sikeres bevezetésének folyamatáról.
Aranyszintű támogatóink: az IBM, az ORACLE és a SUN. E cégek mindegyike 1–1 hangsúlyozottan szakmai tartalmú előadással fog a konferencián megjelenni. Ezüstszintű támogatóink: a Kiskapu Kft., az LSC Kft., a Matáv és az ULX Kft. Bronzfokozatú támogatóink: a Balabit Kft., a Direkt Kft., és a Mission Critical Linux Kft. Fő médiatámogatónk a ComputerWorld Számítástechnika, médiatámogatónk a PRIM.
A nap folyamán ezüst- és bronzfokozatú támogatóink kiállítói standjukon Linux vonatkozású termékek és kiadványok széles skálájával jelentkeznek. Várhatóan négy helyszínen öt időpontban összesen 16 előadás kerül megrendezésre az alábbi témákban:
Bálint Sándor: Linux és a Common Criteria
Baráth Gábor: Megjelenésfüggetlen dokumentáció készítése szabad szoftverekkel
Bodnár Csaba: openMosix: Live free() or die()
Bodnár Csaba: Mitől megy a villamos, avagy a HA-telepekben használt technológiák
Deim Ágoston: Különleges levelezőkiszolgálók
Deim Ágoston: Hálózatfelügyeleti megoldások Linux-alapon
Dr. Mlinarics József: Hazai pályázatok és a magyar részvételi lehetőségek az EU programjaiban
Fejős Tamás: Csoportmunka, Együttműködés, Projekttámogatás linuxos eszközökkel
Kabai József: Az SQL-ledger
Kadlecsik József: Linux alapú tűzfalak Gbit-es környezetben
Keszei Csaba: Biztonságos hálózati elérés vezeték nélküli kapcsolat felett
László Gábor: Nyílt forráskódú szoftverek és a kormányzatok – Egy kutatás tapasztalatai
Nagy Bence: Scribus – DTP Linux alatt
Nemes Dániel: A tűzfalon túl – A kéretlen levelek szűrésétől a teljes tartalomszűrésig
Németh László: A Szószablya projekt
Noll János: Az OpenOffice.org múltja és jövője
Tomka Gergely: Információs forradalom
Tóth Sándor: Tartalomkezelő Rendszer(ek) TYPO3
Verók István: FSRv2 <> : Fordítást Segítő Rendszer (v2)
Vomberg István: Felhasználói programok fejlesztése Gnome/GTK környezetben a Glade fejlesztőrendszer segítségével
A végleges előadások és az időbeosztás a magazin megjelenésének időpontjában a konferencia honlapján (➲ http://konf2003.linux.hu/) már megtekinthető lesz.
Mivel mostanra elfogytak az ingyenes, azaz a támogatott belépők, felhívjuk kedves leendő vendégeink figyelmét az üzleti belépőkre. Az ilyen belépővel érkező vendégeink a következő szolgáltatásainkat vehetik igénybe:

- a konferenciakiadvány egy példánya,
- az összes előadás bemutatóinak kinyomtatott, egybefűzött példánya,
- a konferencia logójával ellátott, galléros póló,
- névre szóló kitűző,
- a büfé ingyenes használata (kávé, üdítő, sütemény),
- üzleti ebéd a hotel éttermében,
- reklámanyagok, jegyzettömb, toll,
- és a konferencia zárásakor értékes ajándékot sorsolunk ki az üzleti belépőt igénybe vevők között.

Üzleti résztvevőinknek érvényes, visszaigazolt regisztráció után átutalásos számlát küld az LME, nyolcnapos fizetési határidővel. Ajánlott az előzetes regisztráció a ➲ http://konf2003.linux.hu/registration.php oldalon, ugyanis a helyszínen kizárólag üzleti belépő vásárolható.

Nemzeti Szabad Szoftver Stratégia – MITS Központi Kiemelt Program

Szervác Attila, az LME egyik legaktívabb tagja jelenti: az Informatikai és Hírközlési Minisztérium és az Információs Társadalom Koordinációs Tárcaközi Bizottsága megkezdte a Magyar Információs Társadalom Stratégia végrehajtásával kapcsolatos koordinációs munkáját. Ennek keretében megindult a Központi Kiemelt Programok programfüzeteinek a kidolgozása, amely a MITS fő irányait fedi le.
A MITS beavatkozási területeinek két fontos iránya a közcélú, közhasznú adatok, szabványok és a programeszközök biztosítása; valamint a hozzáférés, illetve elérés fejlesztése. Ez az a két terület, ahol alapvető szerepet szán a Szabad Szoftvereknek, így a programfüzetek egyik Kiemelt Központú Programja a Nemzeti Szabad Szoftver Stratégia.
Rendkívül megfeszített munka után a Stratégia legfontosabb pontjait 2003. október 15-én vitatta meg a programfüzeteket összeállító Információs Társadalom- és Trendkutató Központ (ITTK). Ezzel kezdetét vette a Nemzeti Szabad Szoftver Stratégia élete, ez fogja eldönteni, hogy a következő három évben milyen ütemben lehet kidolgozni és megvalósítani egy szabad, demokratikus, az állampolgárok alapvető jogait, és a személyiségi jogokat teljes mértékben tiszteletben tartó, gazdaságilag erős információs társadalmat, amely az Európai Modellnek megfelelően szabad programokra épül.

Gibizer Tibor (gibzo@linuxportal.hu)
Újságíró, immár nyolc éve a Linux elkötelezett híve. Imádja a kutyákat, a kerékpározást és az autós csavargást.

Fsf.hu események

Tavaly az olvasók már megszokhatták, hogy az FSF.hu Alapítvány kisebb-nagyobb bejelentésekkel, fordítóhétvégékkel tűnt fel a különböző médiumokban. A Linuxvilág segítségével a szabad programok iránt érdeklődőket ezentúl havonta szeretnénk tájékoztatni a fontosabb eseményekről.

Azok számára, akik még nem ismerik az FSF.hu Alapítványt, elmondom, hogy tavalyi év közepe táján egy baráti társaságból alakult ki, azzal a célkitűzéssel, hogy a lehetőségeinkhez mérten a szabad programok magyarországi helyzetét megpróbáljuk előmozdítani. Mivel hivatalosan csak majd egy évre rá jegyezték be az alapítványt a budapesti bíróságon, elmondhatjuk, hogy ez a szervezet még igen fiatal. Nem kapkodtuk el a dolgokat, aminek a legfőbb oka az volt, hogy nem szerettünk volna egy újabb civil szervezetet csupán nyúlfarknyi időre összedobni, hanem egy igazán ütőképes és összeszokott csapatot akartunk összekovácsolni, akiknek a bírósági papír csupán kötelező formaság. A csapat rohamosan bővült az elhíresült OpenOffice.org három napos fordítóhétvégén és a későbbi Mozilla- és OpenOffice.org-lektorálópartikon. Ma már számtalan közösségi munkát végzünk, amelyek középpontjában a szabad programok állnak:

- Fordítási, lektorálási munkák: az FSF.hu Alapítvány több teljes fordító- és lektorálóhétvégét szervezett, amelyeknek gyümölcseként az OpenOffice.org és a Mozilla is magyar köntösbe bújt.
- Országjáró felvilágosító kampány a szabad programokról: idén kezdtünk – eleinte a BSA agresszív kampányára való válaszként – ellátogatni az olyan megyékbe, településekre, ahol az előadásainkat igényelték. Mára ez olyannyira kinőtte magát, hogy jelenleg már egyetemek, főiskolák, teleházak hívnak minket előadni. Ebben az évben Egerben, Veszprémben, Békéscsabán, Hajdúnánáson és Sopron városában jártunk. Jelenleg a csapat Erdélybe (Kolozsvárra) készül, ahol a Protestáns Teológiai Intézet diákjai hallgathatják meg előadásainkat az alapoktól az áttérési nehézségekig. Az előadások minden esetben ingyenesek, ugyanis adományokból finanszírozzuk őket.
- A szabad programok hívői számára szervezett kirándulások: ez idáig három alkalommal mentünk kirándulni, ezek kifejezetten a szabad programok barátainak lettek meghirdetve, azzal a nem titkolt szándékkal, hogy a jelentkezőket a természet lágy ölén megismertessük egymással.
- Magyar Linux Dokumentációs projekt: a tldp magyarra fordítását végző csapat rendületlen erővel dolgozik rajta, hogy az informatikával foglalkozó szakemberek magyarul is hozzáférhessenek ehhez a kimeríthetetlen tudásanyaghoz.
- FSF.hu-hírlevél: a visszajelzések alapján elindítottunk egy olyan havi hírlevelet, amely a fenti eseményekről számol be részletesen.

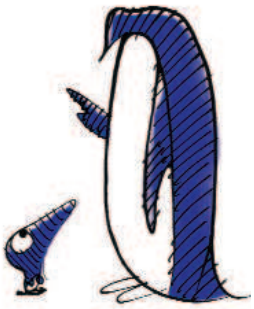
Fiatal szervezetről lévén szó, a jelmondatunk az, hogy csak annyit vállalunk, amennyit minden körülmények között véghez tudunk vinni. Eddigi sikereinket aktivistáinknak köszönhetjük, akik töretlen kedvvel viszik előre a szabad programok ügyét. Aktivista bárki lehet, aki eljön fordítani vagy bármilyen más módon segíti az FSF.hu munkáját, ezzel a szabad programok ismertségét és népszerűségét.

Kapcsolódó címek

Fordítások
➲ http://mozilla.fsf.hu
➲ http://openoffice.fsf.hu
Szabad Szoftver Kampánykörút
➲ http://www.fsf.hu/index.php/Roadshow
Kirándulások
➲ http://www.fsf.hu/index.php/Kirandulas
Magyar Linux Dokumentációs projekt
➲ http://tldp.fsf.hu/
Támogasd te is a munkánkat!
➲ http://www.fsf.hu/index.php/SegitsegHogyan

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

A hónap szakmai tanácsai



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

Rendszerindításkor hiányzik az init!

A Mandrake a következő hibát jelzi a Linux indításakor:

```
mount:error 22 mounting ext2 flags
Kernel panic:no init found.
Try passing init=option to kernel
```

Már háromszor sikerült elindítanom a gépet, de ettől kezdve a rendszerindítás sikertelen, és a fenti üzenetet kapom. Mi a teendő?
Devi, deenadev_20@hotmail.com

A Linux-rendszermag befűzi a gyökérfájlrendszert, és megpróbálja futtatni az `init` programot, amely általában a */sbin* könyvtárban csücsül. A rendszermag forrásából kiolvashatod, hogy melyik 3–4 másik helyen keresi még a Linux az `init` nevű programot. A betöltő parancssorában a teljes elérési úttal te is megadhatod a rendszermagnak az először elindítandó programot (például `init=/bin/sh`). Ezt próbálja a hibaüzenet második része közölni veled. Felvetődik a kérdés, hogy miért nem találja a rendszermag az `init` programot? A legvalószínűbb ok az lehet, hogy nem a gyökérfájlrendszert fűzöd be, hanem egy másikat. Ha a rendszerindításkor egy olyan eszköznevet (például `root=/dev/hda6`) adsz át a rendszermagnak, amelyen egy érvényes fájlrendszer van, de nincs rajta a */sbin/init*, akkor pontosan ezt a hibaüzenetet kapod. Ha a */dev* könyvtár sincs rajta, akkor egy másik hibaüzenet is megjelenik: *Warning: Unable to open initial console.* Ez nem végzetes hiba, talán ez is megjelent, csak nem vetted észre.
Amennyiben hibás gyökérfájlrendszert adsz meg, amelyen olyan fájlrendszer található, amelyet a Linux nem támogat, másfajta hibaüzenetet kapsz: *VFS Kernel panic. Unable to mount root.*
Az is egy lehetőség, hogy a rendszerrel nincs semmi baj, csak véletlenül letörölted vagy átnevezted a */sbin/init* állományt vagy a */sbin* könyvtárat, illetve a fájlrendszer sérült, és nem érhető el a */sbin/init* könyvtárbejegyzése vagy fájlleírója (inode).
Bármi is történt, indítsd el a rendszert a helyreállítólemezről, esetleg egy rendszerindításra képes névjegykártyáról (BBC), amit például a ➲ http://www.lnx-bbc.org webhelyről lehet letölteni. Égesd egy CD-re – általában névjegykártya méretűre szokták, de a hagyományos is megteszi. Indítsd a rendszert BBC-ről vagy Tom *Root/Boot* lemezéről (➲ http://www.toms.net/rb), és minden fájlrendszeren futtasd az `fsck`-t. Ezután nézd végig őket, és állapítsd meg, hogy melyik a gyökérfájlrendszer. Majd próbáld meg kézzel indítani a rendszert a `root=` és az `init=` értékeknek a betöltőnek (LILO vagy Grub) való átadásával. Végül próbáld meg kijavítani a hibás bejegyzést a */boot/grub/menu.lst* (*grub.conf*) vagy a */etc/lilo.conf* fájlban (feltéve, hogy ez okozta a gondot). Amikor a rendszer a BBC vagy más helyreállítólemez segítségével már felállt, jó ötlet megkérdezni magadtól, hogy nem kellene-e menteni az adatokat ezekről a fájlrendszerekről. A legrosszabb esetben újratelepíted a gépet. Nincs abban szégyellnivaló, ha azt mondod: „Az adataimról van mentés, egy szót sem értek az `fsck`-ból és a betöltő beállításaiból, inkább újratelepítek mindent!"
Jim Dennis, jimd@starshine.org

Helyi tartománynév?

Amikor kis otthoni hálózatot (Red Hat 9) építünk, amelynek az egyetlen kapcsolata az internettel egy tárcsázós PPP-kapcsolat, mi a legjobb megoldás a tartománynév választására, hogy az ne ütközzön valamilyen bejegyzett tartománynévvel, amikor az internetre csatlakozunk? Van valamilyen szabványos tartománynév, ami olyan, mint a 127.0.0.1-hez tartozó helyi gépnév?
Steve Cavender, SyntheSys@csi.com

Nincs szükséged névre, az internetkapcsolat enélkül is működni fog. Ha tényleg szeretnél tartománynevet, jegyeztess be egyet.
Usman Ansari, uansari@yahoo.com

Az RFC 2606 kísérleti célokra tartja fenn a *.test* felső szintű tartományt. Az interneten sohasem fognak *.test* végződésű tartománynevet használni. A másik lehetőség, hogyha nem akarsz fizetni a tartományért, megkéred az egyik, bejegyzett tartománynévvel rendelkező barátodat, hogy adjon neked egy altartományt.
Don Marti, dmarti@ssc.com

A Red Hat a *localdomain* tartományt használja (a gép alapértelmezett neve *localhost.localdomain*).
Marc Merlin, marc_bts@google.com

A minicom nem pppd

Gondjaim vannak a hálózatra való csatlakozással. A `minicom` programot használom, és PPP van beállítva (Red Hat 6.1 az operációs rendszerem). Amikor a `minicom` programmal tárcsázom, a kapcsolat létrejön. Ezután a program nevet és jelszót kér, majd miután beírom őket, mindenféle karakter jelenik meg, de egy percen belül a kapcsolat megszakad. Mit csinálok rosszul?
Chris, calf@lanset.com

Semmi rosszat nem tettél, csak nem a megfelelő programot használtad a feladatra. A `minicom` egy terminálprogram, nem lehet vele PPP-kapcsolatot létesíteni, hogy a távoli hálózat ügyfelévé válj. A `pppd` programot kell erre használnod, amely minden bizonnyal része a terjesztésednek. Gondokat okozhat viszont az, hogy a Red Hat 6.1 eléggé elavult. Lehet, hogy nem sikerül kapcsolódni az internetszolgáltatódhoz, ha túl régi `pppd`-t használsz. Célszerű a `pppd`-nek egy újabb változatát telepíteni, vagy az egész terjesztést frissíteni – amelyik kényelmesebb.
Chad Robinson, crobinson@rfgonline.com

Próbáld ki a grafikus `kpppd`-t, vagy valami hasonlót. Esetleg nem ártana frissíteni a terjesztésedet is, ma már a Red Hat 9-nél tart a világ. Kellemes meglepetés fog érni.
Usman Ansari, uansari@yahoo.com

Új termékek

Meteor Laptop

A Sharp Actius MM10-en alapuló Meteor hordozható számítógép mindössze 1 kg tömegű és másfél centiméter vastag. A Meteor MM10-et előre telepített Linuxszal szállítják. A Linux támogatja a megjelenítőt, a hangkártyát, az USB-t, a PCMCIA-t, a vezeték nélküli hálózati eszközt, a hálózatot stb. A gép 10,4"-os, aktív XGA TFT képernyővel van felszerelve, amely az X-et 1024×768-as felbontásban futtatja. A grafikus kártya típusa SMI Lynx. További alkatrészadatok: 1000 MHz-es Crusoe processzor, 512 KB L2-gyorsítótár, 256 MB RAM, beépített 15 GB merevlemez, egy II-es típusú PCMCIA bővítőhely és két USB bővítőhely. Az USB-bölcső lehetővé teszi, hogy az MM10-et kikapcsolt állapotban külső USB-s merevlemezként használjuk. Egy másik linuxos gép állományrendszerébe az MM10 merevlemeze – például adategyeztetés céljából – befűzhető. A Meteorra EmperorLinux, Slackware vagy Mandrake telepíthető, de a Microsoft Windows XP is választható a Linux mellé. A géphez egy hordtáska, egy év támogatás és sok kiegészítő lehetőség jár (az USA-ban).
➲ http://www.emperorlinux.com

OpenPDA Development Studio

Az AMD és a Metrowerks közös munkája az OpenPDA Development Studio az AMD Alchemy Solutions DBAu1100-hoz, amely kis fogyasztású, nagy teljesítményű kézi számítógépek készítését teszi lehetővé. AZ OpenPDA Development Studiónak a program, az Alchemy processzor, a memória és a többi alkatrész egyetlen kártyára építve együtt alkotják a részeit. A MIPS-megfelelő processzor akár 500 MHz-es sebességgel is képes működni 0,5 Watt fogyasztás mellett. Az OpenPDA programja és vasa játékra, zene- és mozgóképlejátszásra, böngészésre és Java-alkalmazások futtatására alkalmas, de egy személyesadat-kezelő program is mellékelve van hozzá. A csomagnak ugyancsak része a beágyazott Linux-rendszermag, a Qtopia, a Jeode JVM és az Opera.
e-mail: sales@metrowerks.com,
➲ http://www.metrowerks.com

Magnia SG30 vezeték nélküli kiszolgáló

A Toshiba SG30 vezeték nélküli kiszolgáló munkacsoportok számára teszi lehetővé a gyors és egyszerű fájl-, nyomtatómegosztást és internetelérést. A Magniát kis- és közepes vállalkozásoknak, illetve fiókirodáknak ajánlják. A noteszgép méretű Magnia lehetővé teszi az adatok megosztását az irodában és azon kívül. A gép képességei: internetátjáró, nyomtatásvezérlő, web gyorstárazása, tartalomszűrés, biztonsági mentés lehetősége, elektronikus levelezés és távfelügyelet, valamint 802.11b IEEE Wi-Fi. Három PCMCIA bővítőhely áll a felhasználó rendelkezésére, támogatja a vezeték nélküli PC card formátumú hálózati kártyákat. További jellemzők: beépített vezeték nélküli elérési pont, Celeron processzor 256 KB L2-gyorsítótárral, 256 MB memória, 128 bites Wi-Fi-titkosítás, beállítható tűzfal és VPN, legfeljebb 160 GB háttértár és két 2,5"-os EIDE meghajtóhely.

➲ http://www.toshiba.com

SpamPlug

A Mail-Filters.com Inc. SpamPlug nevű terméke a kimenő levélszemét ellen vethető be. A program a SpamCure nevű, bejövő levelekre alkalmazható szűrőn alapul, és az internet felé menő levélforgalmat figyeli meg. A levélszemetet eldobja vagy későbbi ellenőrzés végett visszatartja, míg a hasznos levelek azonnal továbbítódnak. A SpamPlug folyamatosan frissített levélszemétminta-adatbázist használ a levélszemét felismerésére, így nő a valószínűsége annak, hogy tényleg a szemetet dobja ki, és nem a hasznos üzeneteket. A SpamPlug gyorsan telepíthető a külön az erre a feladatra kijelölt kiszolgálóra vagy a levélkiszolgálóra.
➲ http://www.mail-filters.com

PCS-620 vezeték nélküli egykártyás számítógép

Az Octagon Systems új egykártyás számítógépe vezeték nélküli elérést, ethernetet, soros és ipari I/O-t tartalmaz, és Linuxot futtat. A kártya legfeljebb 512 MB SDRAM-ot támogat, valamint a CompactFlash bővítőhelyen keresztül legfeljebb 2 GB flashmemória csatlakoztatható hozzá. A PCS-620-ba kis fogyasztású 300 MHz-es Pentium processzort, két 10/100BaseT ethernetkaput, négy soros kaput, RS-233/444/485-öt, CardBus bővítőhelyet a vezeték nélküli, GPS vagy más eszközök számára, két USB-kaput, AC97-megfelelő hangvezérlőt, képcsöves és lapos képernyőhöz való megjelenítőt (legfeljebb 1280×1024), párhuzamos hajlékonylemez és EIDE-kapukat, valamint környezeti hőmérséklet-érzékelő áramkört építettek be. A CompactFlash nem igényel külön illesztőprogramot, és a legtöbb operációs rendszerrel használható.
➲ http://www.octagonsystems.com

QSA

A Trolltech új QSA (Qt Script for Applications) eszköztára lehetővé teszi a Qt-alkalmazások parancsfájlokból történő vezérlését. Pár sort kell csak hozzáadni a Qt-alkalmazáshoz, és lehetségessé válik a kiterjesztése és a vezérlése. A QSA négy összetevőből áll: a QSA SDK az alkalmazásokat a parancsfájlokból vezérlésre felkészítő csomag. A Qt Script több felületen működőképes értelmezett parancsnyelv. A QSA Workbench a parancsfájlok készítését segítő környezet, végül pedig az Input Dialog Framework párbeszédablakok írásához való grafikus programozói felület. Ezzel a csomaggal a fejlesztők már lefordított alkalmazásokat módosíthatnak, tesztelőprogramot készíthetnek az alkalmazáshoz, testreszabhatják azt, parancsfájlon alapuló javítást tehetnek közzé a már megjelent alkalmazáshoz stb.
telefon: 650-813-1676,
➲ http://www.trolltech.com

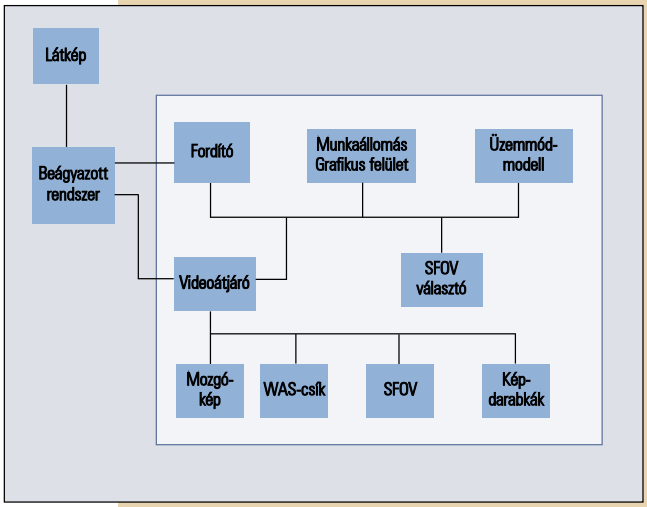
Linux Journal 2003, 113. szám

RSTA-MEP és a Linux-munkaállomás

Önműködően észleli az ellenséget a sötétben, és tájékoztatja a saját erőket annak helyéről.

Nemrég fejeztük be a Linuxra épülő Felderítő, megfigyelő és célbefogó küldetés felszereléscsomag (Reconnaissance, Surveillance and Target Acquisition Mission Equipment Package, azaz RSTA-MEP) munkaállomásának a kifejlesztését. Írásunkban röviden bemutatjuk a teljes rendszert, majd a munkaállomást részletesebben is tárgyaljuk.

A Raytheon RSTA-MEP programja módot ad a harctéri helyzet gyors értékelésére, amit az egyesített fedélzeti és nem fedélzeti érzékelők által nyújtott valós idejű adatok tesznek lehetővé. Az érzékelők és a programok fejlődése lehetővé teszi a széles területű keresésen (wide-area-search – WAS) alapuló képalkotást és az önműködő célészlelést (Automatic Target Detection – ATD), valamint a támogatott célfelismerést (Aided Target Recognition – AiTR). Ezek a képességek a kezelőszemélyzetet valós idejű adatokkal látják el, beleértve a cél helyzetét, osztályozását és elsőbbségének meghatározását. Ezt az amerikai hadsereg harcászati hálózatával (US Army's Tactical Internet) kombinálva lehetővé teszi a kezelők számára, hogy részt vegyenek a saját és az ellenséges erőkről alkotott átfogó hadműveleti kép kialakításában. Ez a jármű egy technológiai bemutató eszköz, amely azt szemlélteti, hogy milyen új képességekkel lehet felruházni a jelen és a jövő felderítő járműveit.



A munkaállomás számítógépes kapcsolatai és moduljai

A jármű jelenlegi kialakításában hőkép-előállító, éjjellátó berendezéssel van ellátva. Az elsődleges érzékelők az árbocon egy nagy hatótávolságú előrenéző infravörös érzékelőnek (Forward Looking Infrared – FLIR), egy inerciális (tehetetlenségi) navigációs rendszernek (Inertial Navigation System – INS) és egy helymeghatározó rendszernek (Global Positioning System – GPS) a vevői, ezen felül számos Raytheon NighSight éjjellátó infravörös érzékelő van a járműre erősítve, hogy a kezelőszemélyzet többi tagja is figyelemmel tudja kísérni a jármű közvetlen környékét.

Az árboc négy méter magas, ehhez adódik még a jármű magassága, így összesen öt méterre lehet kitolni. A járműben három főnyi személyzet utazik: a vezető, a parancsnok és a megfigyelőkezelő. A vezető szintén használni tudja az éjjellátó érzékelőket, hogy éjszaka, sötétben is tudjon vezetni és biztonsági okokból körbe tudjon nézni. A parancsnok feladata a harcászati hálózat kapcsolatának fenntartása és a két másik kezelő irányítása, valamint ő is hozzáfér az NightSight éjjellátó érzékelőkhöz. A megfigyelőkezelő használja a Linux-munkaállomást, amivel az árbocon lévő érzékelőket és a hozzájuk tartozó beágyazott rendszereket kezeli.

Az RSTA-MEP rendszer az árbocra telepített érzékelőkből, a beágyazott számítógépekből és a Linuxot futtató PC-s munkaállomásból áll, és mindez egy H1 Hummer terepjáróra lett telepítve. A részegységek az *ábrán* látható módon kapcsolódnak egymáshoz.

A beágyazott oldal

A beágyazott számítógépek digitális jelfeldolgozó processzorok, amelyek az érzékelők mechanikáját és elektronikáját vezérlik (például beirányozzák vagy lehűtik a detektort), emellett a képfeldolgozásban is részt vesznek. Ezek VxWorksöt futtató PowerPC-kártyák, Microsoft Windows NT-t és Sun Solarist futtató egykártyás gépek. A rajtuk futó alkalmazások között megtalálható a Force XXI Battle Command Brigade and Below (FBCB2), ami egy amerikai katonai digitális utasító- és irányítórendszer, célkereső és azonosító, valamint képfeldolgozó és kapcsolattartó program is. A csomag tartalmaz egy GPS-vevőt, tehetetlenségi navigációs rendszert és digitálistérkép-szolgáltatást. A beágyazott rendszerek optikai kábelen keresztül, ethernet és Virtual Interface (VI) protokoll segítségével tartják egymással a kapcsolatot.

Kapcsolódás a munkaállomáshoz

A Linux-munkaállomás prototípusa egy már létező rendszer utóda. Az eszményi eset az lett volna, ha munkaállomásunk pontosan beillett volna az előző helyére. A VI és az optikai kábel használatára irányuló első kísérleteink kudarcot vallottak. A beágyazott rendszerekkel foglalkozó csoportunknak már tekintélyes tapasztalataik voltak a beszállítók együttműködő-képességéről, vagyis az optikai kábelt választva csupán azokra a beszállítókra hagyatkozhattunk, akik VxWorks és Linux alatt működő kártyákat és meghajtóprogramokat egyaránt tudtak kínálni. Ezek közül egyet sem találtunk, aki a VI protokollt támogatta volna. Második kísérletként megpróbáltunk lemezemulációt használni és annak a látszatát kelteni, hogy egy merevlemezt csatlakoztattunk az optikai kábelre, így legalább ugyanazon az adathordozón belül maradhattunk.

Az eredmény szintén nem volt kielégítő, így áttértünk gigabit ehternetre. Az ethernet tudná hordozni egyrészt a videójelet az érzékelőktől, másrészt a parancs- és állapotadatokat a munkaállomás és a beágyazott rendszerek között. Gigabit ethernet megoldás keresésekor négy dolgot kell meghatároznunk: a csomagméretet,

az átviteli közeget, az összekapcsolás módját és a hálózati csatolókártyát.
A hagyományos ethernet legnagyobb csomagmérete 1500 bájt. Az egyre inkább terjedő gigabit ethernet 9000 bájtnyi legnagyobb csomagméretet engedélyez, ezt nevezik „jumbo" csomagnak. A mi projektünkben a beágyazott oldal és a linuxos oldal közötti megfelelőség miatt a hagyományos csomagméret mellett döntöttünk.
A következő, amit meg kell fontolnunk, a média. A gigabit ethernetkártyák két változatban kaphatók: réz- és optikai kábellel. A réz az elektromágneses interferenciára (EMI) érzékeny, míg az optikai kábel mechanikailag sérülékeny. Végül az ára miatt a rezet választottuk. Ha az elektromágneses interferencia gondot okozna, bármikor optikai kártyára lehet váltani a program módosítása nélkül. Azért is választottuk a rezet, mert könnyen tudtuk illeszteni laboratóriumunk meglévő berendezéseihez. Meglévő hálózatunk (a 10/100-as) is rézvezetékes volt.
A harmadik szempont az összekapcsolás módja. A helyzet nem sokat változott a 10/100-as ethernethez képest: vannak kapcsolók (switch), jelelosztók (hub) és átkötő kábelek. A kapcsolók irányítják a forgalmat, amit így csak a címzett kap meg. Kezelik a különböző sebességű, egy- és kétirányú kapcsolatokat, és villogó fényekkel jelzik a működést, segítve a hibakeresést. A kapcsolók hátulütői az áruk, és hogy felügyelhető hálózati kapcsolóra van szükség, ha csomagfigyelőt (packet sniffer) szeretnénk használni.
A másik lehetőség a jelelosztók használata. Előnyük, hogy olcsóbbak, mint a kapcsolók, és nekik is vannak állapotjelző lámpácskáik. A gond az, hogy eddig nem találkoztunk gigabit ethernetes jelelosztóval (csak kapcsolóval), ha pedig 10/100-as jelelosztót használunk, akkor fel kell áldoznunk a sebességet. A jelelosztók ráadásul az összes csomagot minden vonalra elküldik, ami jó, ha meg akarjuk figyelni a csomagokat, de rossz, ha korlátozni akarjuk a csatolón átfolyó forgalom mennyiségét.
A legegyszerűbb megoldás az átkötőkábel használata. Ez a legolcsóbb, nem igényel más eszközt és biztosak lehetünk benne, hogy külső forrásból nem érkezik csomag. Az is igaz viszont, hogy nincsenek villogó lámpácskák, nincs mód kívülről megfigyelni a csomagokat, és ha egy csatoló leáll (például újraindul a beágyazott gép), akkor megáll a másik is.
Végül a kapcsolók mellett döntöttünk, bár a kapcsolók és átkötőkábelek közötti választás még mindig „vallási" vita tárgyát képezi. Szintén nagy gondot fordítottunk a gigabit-kábelezésre. A profi módon előállított CAT 5e és CAT 6 kábelek használata előnyösebb a házilag készített kábeleknél.
A negyedik eldöntendő kérdés a hálózati csatolókártya választása. Ezek 32 vagy 64 bitesek lehetnek. A 64 bites kártyák jellemzően jobb teljesítményt nyújtanak és kevésbé terhelik le a PCI-sín erőforrásait. Bár nem végeztünk piackutatást az elérhető termékek körében, az Intel Pro/1000 Server Adapterre esett a választásunk.
A protokollok közül a TCP/IP mellett döntöttünk.

Bár a TCP lassabb, mint az UDP, megbízható, kijavítja az eldobott, kétszerezett vagy nem sorrendben érkező csomagok által okozott hibát. Szerettük volna a legjobb videóminőséget elérni a lehetséges elektromágneses interferencia ellenére is, ezért úgy véltük, hogy a beépített hibajavítás megléte alapvető. Ezenkívül az sem elhanyagolható, hogy így az utasító- és állapotadatok

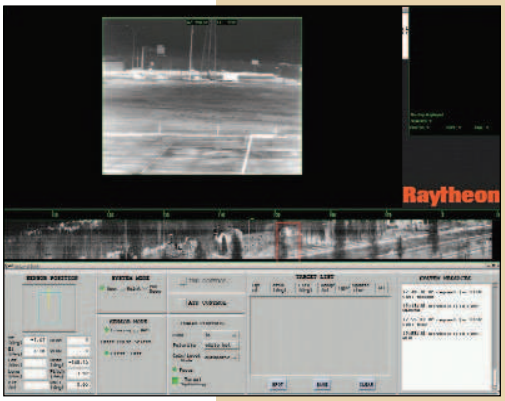
1. kép
Hummerre telepített RSTA-MEP rendszer kitolt árboccal. Az érzékelők az árboc tetején vannak, a beágyazott rendszerek hátul, a fehér dobozokban. A munkaállomás számítógép a jármű belsejében található

is megbízhatók maradnak. Amikor a foglalat- (socket) réteget kódoltuk hozzá, be kellett hangolnunk a foglalat küldő és fogadó átmeneti tárának méretét (a `setsockopt`-ot használtuk a `SOL_SOCKET SO_RCVBUF` és `SO_SNDBUF` kapcsolókkal), hogy az áteresztőképessége elég legyen a videó számára. Ezenkívül kikapcsoltuk a Nagle-algoritmust (a `setsockopt IPPROTO_TCP`-vel és a `TCP_NODELAY`-jel), hogy csökkentsük a késleltetést a munkaállomás és a beágyazott rendszer között, amivel ez utóbbit érzékenyebbé tettük a munkaállomáshoz csatlakoztatott vezérlőkarok által adott érzékelő-irányzó parancsokra.

Munkaállomás-alkalmazói program

Ez a munkaállomás-prototípus a Raytheon Tiger szimulátorból nőtte ki magát – a beágyazott oldallal ellentétben, amely az amerikai hadsereg fegyverek és rendszerek technikai kialakítását végző munkacsoport (Weapons Systems Technical Architecture Working Group – WSTAWG) által kidolgozott általános kezelői környezet (Common Operating Environment – COE) átültetése.
Bár mind a munkaállomás, mind a beágyazott oldal rendszere üzenettovábbító alapú, ezek egymással nem működnek együtt. Ezért egy fordítómodult szükséges a kettő közé illeszteni. A késleltetés és a processzorterhelés csökkentésére ez a fordítófolyamat (process) két szálra van szétválasztva, a Posix-szálak (threads) függvénykönyvtárának a használatával. Az egyik szál a beágyazott oldalon vár üzenetre, és azt lefordítva a munkaállomás elemei által használt megosztott memóriaterületre helyezi. A másik szál ugyaninnen veszi a

beágyazott rendszereknek szóló üzeneteket, hogy fordítás után továbbítsa azok bemenetére. A fordítómunka két szálra osztásával és a program javításával (optimizations) a késleltetést a legkisebb értéken lehet tartani.



2. kép
A munkaállomás mozgó videomódban

A videóátjátszó modul egy, csak ennek a feladatnak szentelt külön gigabit ethernet hálózati kapcsolaton keresztül olvas. Meghatározza, hogy melyik videóablakban kell megjeleníteni a képet, és oda továbbítja.
A munkaállomáson a kezelőfelület vezérlőpanelje a Builder's Xcessoryval lett létrehozva. Három fő szempont vezérelte a kialakítását: a korlátozott képernyőméret, a kész felület tükrözze a beágyazott oldal állapotát, valamint egér vagy hanyatt egér (trackball) helyett vezérlőkart (grip) lehessen használni.
Az első fő tervezési nehézség, amivel szembekerültünk, a képernyő területe volt. Egyetlen monitoron zajlik minden megjelenítés, ezért csak a képernyő alsó harmada áll a kezelőfelület rendelkezésére. A rendszer üzemmódja és ennek az üzemmódnak a vezérlőelemei tehát ebben a harmadban jelennek meg. A rendszernek két fő üzemmódja van: a WAS-mód és a hagyományos kameramód. WAS-módban az érzékelő gyorsan pásztázza a kezelő által meghatározott területet, miközben a vezérlőkarokkal ki lehet választani egy részt, amit szuper látómezőként (Super Field Of View – SFOV) lehet megjeleníteni. Kameramódban élő videókép látható és a vezérlőkarokkal lehet az érzékelőt mozgatni. Egyik üzemmódban sincs szükség a másik mód vezérlőelemeire, ezért két elemkészletet (widgets) terveztünk ugyanarra a képernyőterületre. A kezelőfelület-érzékelő üzemmód panelje a *2.* és *3. képen* látható. Amikor az egyik készletet használjuk, a másik el van rejtve.
Egyéb szolgáltatások, mint például az önműködő célészlelő program kezelőszervei, egy másik ablakban kaptak helyet, és a fő kezelőfelületről egy gombnyomással elérhetők. Ezek az ablakok a képmegjelenítő terület fölé ugranak be.
A képernyőterület szűkössége egy másik kérdést is felvet: szükség van a felhasználói utasításokra válaszoló rendszer azonnali, látható visszajelzésére csakúgy, mint a beágyazott rendszer pillanatnyi állapotának jelzésére. Külön elemek helyett ugyanazokat alkalmaztuk vezérlő- és állapotjelző objektumokra. Amikor a rendszer kezelője használ egy elemet, a rendszer a kiadott parancsát önműködően visszajelzi a kezelőfelületen, közben az elem visszahívó kódja indításra kerül. Ez a kért változással elküld egy üzenetet az üzemmódmodellnek. Ez a kérés átkerül a beágyazott érzékelő oldalra, amely egy állapotjelentést ad vissza. Amennyiben az állapot eltér a kéréstől, az üzemmódmodell értesíti a kezelőfelületet, amely frissíti az elemet, hogy a pillanatnyi állapotértéket mutassa.
A harmadik tervezési nehézséget az egér nélküli környezet iránti igény jelentette. A jármű mozgása és a tényleges munkaasztal hiánya nehézzé teszi az egér, a trackball vagy az érintőképernyő használatát. Egy billentyűzet rendelkezésre áll ugyan, de csekély mennyiségű adatbevitelre használatos. Ebből kifolyólag úgy terveztük, hogy a kezelőfelületet kézi vezérlőkarral lehessen kezelni.
Az egér nélküli módot a kezelőfelület korai változatában kézi elembejárási útvonalak és gombnyomásesemények hozzáadásával értük el. A vezérlőkaron lévő sapkakapcsoló (hat switch) mozgatása az `XmProcessTraversal` hívásával változtatta az elem fókuszálást. A kiválasztó (Select) gomb megnyomása egy, az alábbihoz hasonló `XEvent`-et határozott meg és küldött el:

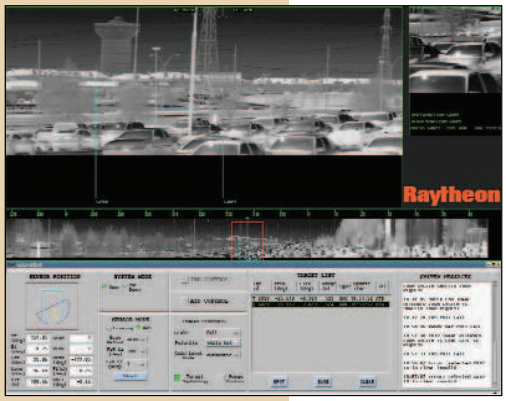
```
/* sending key press events */
#include <X11/keysym.h>

XKeyEvent ev;
Window     rootWin;
int        x,y;
int        root_x,root_y;
Window     win;

rootWin =  RootWindowOfScreen
➥(guiScreen);

win = findPointerWindow(rootWin,
➥&x, &y, &root_x, &root_y);

ev.type = (long) KeyPress;
ev.send_event = True;
ev.display = display;
ev.window = win;
ev.root = rootWin;
ev.subwindow = 0;
ev.time = CurrentTime;
ev.x = 1;
ev.y = 1;
ev.x_root = 1;
ev.y_root = 1;
ev.state = 0;
ev.same_screen = True;

ev.keycode =
➥XKeysymToKeycode(display,XK_space);

XSendEvent(display, window, True,
➥KeyPressMask, (XEvent *)&ev);
```

A kezelőfelület jelenlegi változatától eltérően az előző változat mindössze egy `topLevelShell`-ből állt, amely csak egyszerű elemeket tartalmazott, például ilyen a `PushButtons` és a `ToggleButtons`. A jelenlegi

kezelőfelületben több héj (felugró ablakok) és összetett elemek találhatók, mint például az *OptionMenus*.
Az `XmProcessTraversal` egyszerű meghívása a fókusz megváltoztatására nem működik héjak között. Egy gombnyomást az *OptionMenu*-n elküldve a menü felugrik, mindazonáltal egy második gomb megnyomása nem választ ki egy újabb lehetőséget, és nem tünteti el a menüt.
Néhány jó tanács olvasóinknak – nem árt, ha az alábbiakról nem feledkezünk meg:

1. Az ablakkezelő a főnök. Amikor több héjjal akad dolgunk, emlékezzünk rá, hogy az ablakkezelők nem igazán adják át a fókuszt vagy bármilyen hasonló feladat vezérlését.
2. Az elemhierarchia hatása: egy elemkészlet csoportjában a bejárási út rendjét részben az a sorrend határozza meg, ahogyan a kódban meg lettek adva (declared).
3. Bánjunk óvatosan a színfalak mögötti kóddal! Nézzünk egy `RadioBox`-ot, amely két `ToggleButton` gyermeket tartalmaz, ezek közül az A van kiválasztva. Amikor egy, a B-t kiválasztó üzenet érkezik, a gyermek `XmNset` erőforrások értékeinek egyszerű felcserélése a képernyőn helyesnek látszik. A szülőelem eközben még mindig azt hiszi, hogy az A van kiválasztva, ami nem várt `Button` működéshez vezethet.

Projektünk jelenlegi változatában egy külön folyamat foglalkozik a kézi vezérlőkarról érkező bemenettel, és vezérli az egérmutatót – az `XWarpPointer` és az X-kiszolgáló `XTest` bővítményének kombinációját használva (lásd a *Kapcsolódó címeket*). A munkaállomás a beágyazott oldalról küldött adatokból létrehozott videójelet is megjelenít. A videóátviteli folyamat ezt egy foglalatból olvassa ki és egy ablakba továbbítja. Négy ablak van: mozgó videó, WAS, SFOV és képdarabok.
Mint már említettük, a mozgó videó az élő képadat.
A WAS ablak képe egy összenyomott képcsík, ami egy állandó területet gyorsan pásztázó érzékelőtől érkezik. A WAS-csík jelrendszere többek között jelzi azokat a helyeket, amelyeken a célzó rendszer szerint célpontok találhatók. Az SFOV egy több részletet mutató nagyobb nézet a WAS-csík egy részéről, amit a kezelő választhat ki. Ezen láthatók a célzó szimbólumok és adatok a digitális térképről. A képdarabok a terep azon részeit mutatják, amelyeken a célzórendszer valami érdekeset talált. Ezeket megmutatja a kezelőnek értékelésre és jelenti más, a járművön kívül lévő rendszereknek. A 2. képen egy mozgó videómódban készített külső nézet látható a kezelőfelülettel, a videóablakkal és a WAS-csíkkal.
A 3. képen a rendszer WAS-módban van, látható rajta a WAS-csík, az SFOV ablak, a kezelőfelület és egy képszeletablak.
A videót OpenGL-ben egy poligonon lévő textúraként alakították ki, a videóból érkező adat erre a textúrára kerül. Amikor a poligon képernyőre kerül, akkor a videó is látható (lásd a példakódot az 53. CD Magazin/RSTA könyvtárában ami ezt a technikát szemlélteti). Azért választottuk az OpenGL-t a videóhoz, mert számos lehetőséget kínál az adat feldolgozására és megjelenítésére. A kép átméretezhető vagy elforgatható, ha más tájolásban készült, mint ahogy meg van jelenítve. Az OpenGL számos primitívet tartalmaz a szimbólumok képekre rajzolásához; a villogásmentes frissítéshez képfeldolgozó képessége és kettős átmeneti tárazása révén járul hozzá. Az OpenGL hordozható és jól dokumentált, ráadásul a processzorról egy csomó munkát átirányíthatunk a grafikus kártyára.



3. kép
A munkaállomás WAS-módban

Az SFOV kiválasztója vezérli, hogy a WAS-csík mely része kerüljön megjelenítésre az SFOV ablakban. Ezenkívül azt is szabályozza, hogy a piros négyszög a WAS-csík ablakán belül hova kerüljön kirajzolásra.
A munkaállomás egy külön vezérlő- és módválasztó modullal bír. Ahelyett, hogy a logikai egységek szét lennének szórva a rendszeren belül különböző modulokra, ebben a modulban koncentrálódnak. Ez a kialakítás a rendszer többi részét egyszerűbbé és könnyen újra felhasználhatóvá teszi. Ugyanakkor a módválasztó modul nagyon összetett. A módmodellnek meg kell tudnia valósítania mindazt az ismeretet, hogy miként hatnak egymásra az egységek és hogyan tükrözik a beágyazott rendszer és a munkaállomás állapotát. Lehetővé teszi a munkaállomásnak, hogy az összegyűjtött adatokra alapozva engedélyezett akciókat végezzen, és hogy a beágyazott oldalt hibák és a helyzet váratlan megváltozása után kutatva megfigyelje.

Milyen gyors az elég gyors?

A munkaállomás az éppen valós idejű kategóriába esik: a rendszer nem hibázik, ha valami késik. Mivel ez egy emberi tényezőt is tartalmazó próbarendszer, amelynek a legtöbb időkritikus összetevője a beágyazott rendszerekben rejlik, csak olyan gyorsan kell futnia, hogy a kezelő végre tudja hajtani a feladatokat. Ezért nem használjuk a valós idejű Linux-keretrendszerek egyikét sem, ehelyett nagy teljesítményű alkatrészekkel oldottuk meg a feladatokat: SCSI merevlemezzel, elég memóriával, hogy kiküszöböljük a lapozást, és egy GeForce4 grafikus kártyával a gyors OpenGL miatt, valamint két 2,4 GHz-es processzorral a Microwaytől.
Egy korlát emelkedett a rendszer két része, a videó és az érzékelő célzása elé. Az élő videó RS-170 sebességgel kerül betáplálásra a munkaállomásra, egy félképnyi képsor minden 1/60-ad másodpercben. Ezeket olyan gyorsan kell összerakni, és megjeleníteni, hogy állandó se-

bességet lehessen elérni és fenntartani. Hogy ezt lehetővé tegyük, biztosítottuk, hogy elegendő hálózati sávszélesség legyen a videójel szállítására és elégséges processzor- és grafikus teljesítmény a megjelenítés frissítésére. A monitor frissítését 60 Hz-re beállítva helyben is voltunk (lásd az nVidia meghajtóprogramhoz mellékelt *README.txt* fájlt, a legfrissebb a ➲ http://dowload.nvidia.com/XFree86_40/1.0-4194/README címről tölthető le).

Az érzékelő irányítása hasonló kihívást jelentett. Ennek elég érzékenynek kell lennie a vezérlőkarokra, hogy a kezelő a cél elvesztése nélkül tudjon vele célozni. Míg a videó leginkább a sávszélességtől függ, az érzékelő irányítása a késletetésen múlik. Egy hosszú üzenetlánc késleltetéshez vezet. Például egy gombnyomás vagy a vezérlőkar-kezelőfelületről származó elfordulás parancs eljut a vezérlőfolyamathoz, hogy megállapítsa, melyik bemenet érvényes, majd továbbmegy a fordítóhoz, hogy EO formájú üzenet legyen belőle. Ezután keresztülhalad a gigabit ethernet hálózaton egy beágyazott folyamatba, amely fogadja az üzenetet, majd továbbítja az OE-be és a beágyazott rendszer kódjába, onnan a működtető szerkezetre kerül, végül az eredmény visszaérkezik a videoadatfolyamba. A fordítófolyamatok két szálra választása és a teljes optimalizálással történő fordítás (`-Wall -ansi -O3 -ffast-math -mpentiumpro`) megtette a magáét.

A `gprof` profilert használtuk, hogy megkeressük, vannak-e kényes helyek a kódban (lásd a `gprof` információs oldalát). Itt a videókód profileozása közben komoly nehézségbe futottunk: amikor X-időzítőt használtunk (`XtAppAddTimeOut`), nem került időzítési adat a profileba. (Talán a profiler és az `XtAppAddTimeOut` ugyanazt a jelet használták, és egymást zavarták?) A másik egyszerűsítési lehetőség, amire rájöttünk: a videóforráshoz a páros és a páratlan sorok – két kisebb helyett – egyetlen paranccsal történő továbbítása a hálózaton.

Előnyök, kelepcék és következtetések

A Linux használata néhány gondot is okozott, például nem találtunk olyan beszállítót, aki PCI Mezzanine kártyát tudott volna szállítani PowerPC-hez VxWorks-meghajtóval, vagy PCI kártyákat Linux-meghajtóval, illetve aki kezelni tudta volna a VI protokollt. Végül el kellett vetnünk az optikai kábel használatának ötletét.

Számos alkalommal viszont azt tapasztaltuk, hogy a Linux használata előnyt jelentett. Mivel a rendszert merevlemezről indítottuk, nem kellett EEPROM-ba égetni, mint a beágyazott rendszereknél. Amikor ott EEPROM-ba kerül a kód, sokkal nehezebbé válik a hibakeresés. Ezenkívül a Linux magfájlokat kínál a hibakeresés segítésére, amit a VxWorks nem tesz meg. A Linux-munkaállomás jóval masszívabb felépítésű és jobb képminőséget nyújt, mint elődje. Végül a műhelyben történő összeállítás és az egységek kipróbálása könnyebb Linux alatt, mivel a kereskedelmi forgalomban kapható PC-k jóval elterjedtebbek, mint a beágyazott PowerPC-k. Arra számítunk, hogy a jövőben a Linux, az X és az OpenGL-környezet teljesítménye és rugalmassága egyre kifizetődőbb lesz, ahogy több módot és eszközt adunk prototípusunkhoz.

Linux Journal 2003. október, 114. szám

George Koharchik (g-koharchik@raytheon.com)
A Raytheon's Visualization és Simulation Lab (VSL) munkatársa, szabadidejében a biztosítótű mechanikáján elmélkedik.

Quintelle Griggs
(Quintelle_Y_Griggs@raytheon.com)
A Raytheon's VS munkatársa.

Sonja Gross (sonja_gross@raytheon.com)
2001-től a Raytheon's VSL munkatársa, miután megszerezte baccalaureátusi fokozatát számítástechnikából a Louisiana Tech Universityn.

Kathy Jones (kajones@raytheon.com)
Programfejlesztő a Raytheon's VSL-nél. Motif VAPS kezelői felületeket és egyéb programeszközöket készít.

John Mellby (j-mellby@raytheon.com)
A Raytheon's VSL munkatársa, virtuális szimulációval foglalkozik, rövid biográfiákat ír, és megméri a texasi napot a tavaszi napéjegyenlőség idején.

Joe Osborne (joe.osborne@smiths-aerospace.com)
A Smiths Aerospace munkatársa a Michigan állambeli Grand Rapidsben.

KAPCSOLÓDÓ CÍMEK

BX ➲ http://www.ics.com/products/bxpro
Integrated Computer Solutions ➲ http://www.ics.com
nVidia ➲ http://www.nvidia.com
Microway ➲ http://www.microway.com
Eric F. Johnson és *Kevin Reichard* Power Programming...Motif; Management Information Source, Inc. Második kiadás Version 1.2, 1993., New York. ISBN: 1-55828-319-6.
Raytheon ➲ http://www.raytheon.com
NightSight ➲ http://www.raytheon.com/products/tiger
Nem hivatalos VxWorks és Tornado GYK
➲ http://www.xs4all.nl/~borkhuis/vxworks/vxfaq.html
A VxWorks és a Tornado a Wind River Systems termékei
➲ http://www.windriver.com
VxWorks, illetve Tornado II GYK (különösen a 4.6-os rész a foglalatokról)
➲ http://www.xs4all.nl/~borkhuis/vxworks/vxworks.html
WSTAWG weboldal ➲ http://wstawg.army.mil/index.asp
XTest bővítmények XFree86-hoz
➲ http://xfree86.org/pub/XFree86/4.2.0/doc/xtestlib.TXT

Egy nemzet programok nélkül: Linuxot Iraknak!

Egy felelőtlen, korrupt kormány, a háborúk és a szankciók után Irakban nem létezik programipar, így a Linux tiszta lappal indulhat.

Irak mostanáig a számítógépes programok sötét középkorát élte, a rendelkezésre álló programok és a saját fejlesztések tekintetében is. Miközben a világ többi részében a programhasználat és -fejlesztés óriási fejlődésen ment keresztül, Irak lemaradt. A háborús válságok és a szankciók miatt a legnagyobb intézményeknek – a kormányt is beleértve – nem volt lehetősége az igényeiknek leginkább megfelelő megoldások felkutatására és bevezetésére. Ehelyett a legelső ajánlat mellett döntöttek – sajnos ez nem a Linux volt.

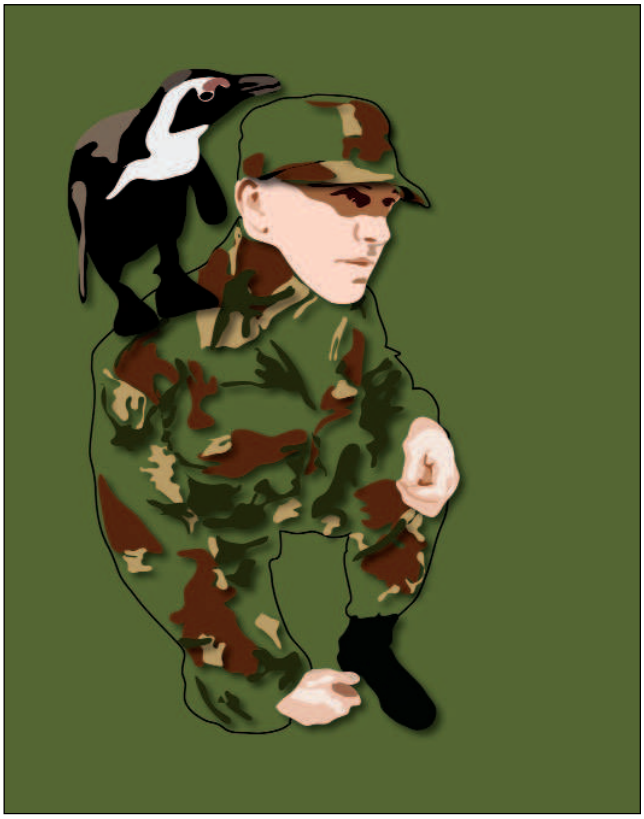
A programok világában a változások követhetetlenül gyorsnak bizonyultak, különösen az olyanok számára, akiknek nem adatott meg az a háttér, ami a hatékony és sikeres programfejlesztés előfeltétele.

Az új iraki helyzetben visszatért a szerzői jog intézménye, és a cégek hajlanak arra, hogy fellépjenek a jogosulatlan felhasználással szemben, vagy legalább megelőzzék további burjánzását.

Ez arra fogja ösztönözni az iraki felhasználókat, hogy a jogdíjas programokat más megoldásokkal helyettesítsék, különösen azért, mivel itt a költségek szerepe nyilvánvalóan jelentős, mégha nem is a legfontosabb szempont.

Noha manapság szinte lehetetlen számítógépek nélkül élni, a korábbi iraki kormány hanyag és korrupt hozzáállása számos kulcsfontosságú területen visszafogta a programbeli megoldások elterjedését, vagy akár visszafejlődést eredményezett. Így aztán, ha időben cselekszünk, a Linux számtalan területen hatalmas segítséget jelenthet a valóban működőképes háttér költségkímélő kialakításában.

Ehhez szükség lehet egy iraki Linux Közösségi Központra, ami a nehezebben alkalmazható megoldások terén segítséget nyújthat az első lépések során. A helyzetet egy fehér vászonhoz is hasonlíthatjuk, amit a legkülönfélébb színekkel boríthatunk be, teljesen szabadon választva a használható színek közül.



Biztos vagyok benne, hogy megszámlálhatatlan jó ötlet vár megvalósításra, amelyekkel bárki élete sok szempontból megkönnyíthető. Mivel infrastruktúráról beszélünk, az oktatás lehet a legalkalmasabb hely a Linux terjesztésére, ami lehetőséget teremthet arra, hogy a Linux bevezetése könnyű, gyors és szakszerű legyen, és hasznosnak bizonyuljon itt, Irakban.

A tudásháló megalapozása vagy újraépítése önmagában is beletelik némi időbe, ami újabb lehetőséget kínál ahhoz, hogy a Linux legyen az egyik sarokköve ennek a folyamatnak – részévé válhat a mindennapi számítógépes munkának. Például több középiskolában az utóbbi évtizedben nem változott a tanterv, és nem oktatnak alapvető számítástechnikai ismereteket sem.

Itt lehetőség nyílhat arra, hogy a Linux-rendszer alapjait ne egy másik rendszerről való áttérés szemszögéből mutassuk be, de még csak nem is egy új operációs rendszerként, hanem olyan hatékony operációs rendszerként, ami lehetővé teszi a választást, még mielőtt a friss elmék bezárkóznának egy másik rendszerbe. Ez magasabb szinten egyes főiskolákra és más intézményekre is igaz.

Nagy munka vár ránk.

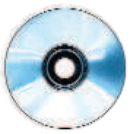
Az új iraki Linux-felhasználói csoportnál az elsődleges célunk egy Linux-oktatólabor és egy -könyvtár alapítása. Felkértük a bagdadi Al Nahrain Egyetemet és a Bagdad Egyetemet, hogy lehetőséget biztosítsanak Linux-tanfolyamok megtartására. Fontos iraki köz- és magánintézmények vezető informatikai szakembereit hívjuk meg linuxos tanfolyamokra és szemináriumokra. Amikor a résztvevők intézményei kialakítják Linux-telepítési tervüket, együtt fogunk működni velük a leírások helyi változatának elkészítésében és a közösségi erőforrások megszervezésében, ideértve a levelezőlistákat és a terjesztések tükrözését is.

Honlapunkon, a ➲ http://linux-iraq.org-on nemsokára megtudhatod, hogyan segíthetsz.

Linux Journal 2003. október, 114. szám

Ashraf T. Hasson
Bagdadban él. A Mérnöki Főiskola Lézer Tanszékének végzős hallgatója. Az új iraki Linux-felhasználói csoport alapítója.

Rendszerindításra képes helyreállító lemezek



Egy rendszert a puszta vasra visszaállítani vagy egy az egyben lemásolni a leggyorsabban egyedi helyreállító CD készítésével lehet. Bemutatjuk e folyamat gépesítését és a helyreállító CD-k naprakészen tartásának a módját.

A Mondo nevű, katasztrófa utáni helyreállító programcsomag létfontosságú része lett a Midwest Tool & Die (MTD) adatmentési tervének. A Mondo biztonsági mentést készítő eszközét, a `mondoarchive` programot hozzáadtuk adatmentési tervünkhöz. E program segítségével a katasztrófa után a puszta vasra vissza lehet állítani a rendszert, gyorsan visszatérhetünk egy ismert, jó beállításhoz, és teljes Linux-rendszereket is könnyen lemásolhatunk. A Mondo képes klónozni az LVM, a RAID, az `ext2`, az `ext3`, a JFS, az XFS, a ReiserFS és a VFAT fájlrendszereket.

Katasztrófa utáni helyreállítás

Az MTD-nél szalagra mentjük a céges és felhasználói adatokat. Több vállalati biztonsági mentési megoldást is kipróbáltunk már, de mindegyiknek gyenge pontja volt a katasztrófa utáni helyreállítás. Régen a puszta vasra történő helyreállítás azzal kezdődött, hogy a Linuxot a terjesztés CD-jéről újra kellett telepíteni. Ezután lehetett telepíteni a szalagos mentéseket beolvasni képes programot, végül következett az adatok és programok helyreállítása a szalagról.

Jelenleg rendszerindító CD-ket készítünk, amelyekről az egész rendszer – illesztőprogramokkal és alkalmazásokkal együtt – helyreállítható. A szalagon tárolt állományok ezután felülírják azokat, amelyek a CD létrehozása után jöttek létre. Amikor a rendszer jelentősen megváltozik, új helyreállító CD-ket hozunk létre a régiek helyett.

Rendszeres biztonsági mentés

Helyreállító lemezlenyomatokat (CD image) létrehozni egyszerű, ezért ez a folyamat megismételhető, amikor a rendszer pillanatfelvételét kell létrehoznunk. A ma használatos nagy merevlemezekre rendszeresen készíthetünk biztonsági mentést a `mondoarchive` programmal. Később egy `cron`-ból futó megoldást mutatunk be erre a feladatra.

Az MDT-nél bizonyos rendszerekhez a `mondoarchive` programot használjuk a szalagos mentés helyett:

- Az időben változatlan adatokat kiszolgáló Linux-gépekhez felesleges a szalagos mentés. A `mondoarchive` programmal adott időközönként biztonsági mentést készítünk.
- Az internetre közvetlenül csatlakozó számítógépek a tűzfalon kívül vannak, elkülönítve őket a belső hálózattól. A szalagos mentést végző kiszolgáló nem látja ezeket a számítógépeket. A `mondoarchive` segítségével e rendszerek helyesen működő változatát tároljuk. Ha betörést észlelünk, hamar vissza lehet állítani az eredeti állapotot.
- A Linux-tűzfalak és -útválasztók állandó beállításokkal üzemelnek, nincsenek felhasználói fiókok és a biztonsági szint magas, emiatt nem érhetők el a vállalati szalagos kiszolgáló számára. Mivel a tűzfalak beállításai ritkán változnak, a `mondoarchive` jól használható.

Egyéb alkalmazások

A biztonsági mentések tárolása mellett a helyreállító CD úgy is felfogható, mint egy rendszerindításra képes pillanatfelvétel a számítógép állapotáról. A `mondoarchive` könnyű használhatóságának köszönhetően több alkalmazási terület is született vállalatunknál.

A `mondoarchive` a kísérletezést is jól támogatja. A rendszert rátöltjük a puszta vasra, állítgatjuk, és az ismert, jó beállítások bármikor visszatölthetők. Ha a kísérletek eredményesen zárultak le, új lemezlenyomatot (CD image) készíthetünk. Ez lesz az élesben működő kiszolgálónak a katasztrófa utáni helyreállítólemeze.

A Mondo helyreállító segédeszköze rugalmas. Helyreállításkor a lemezrészek átméretezhetők és átrendezhetők, ami akkor hasznos, ha például egy nagyobb merevlemezt szerelünk be a régi helyére a számítógépben.

Teljes Linux-rendszereket is lemásolhatunk, ha a helyreállító CD-re mindent felteszünk. Ezután egy üres gépre visszatölthető az egész. Ha az eredeti rendszer DHCP-t használt az IP-cím megszerzésére, akkor az új rendszer is így viselkedik majd. Az állandó IP-címeket használó rendszerek másolásakor a másolat ugyanazzal az IP-címmel és gépnévvel fog rendelkezni. Vigyázzunk arra, hogy ez a másolat ne legyen csatlakoztatva a cég éles hálózatához. Mi egy külön kísérleti hálózatot tartunk fenn erre a célra.

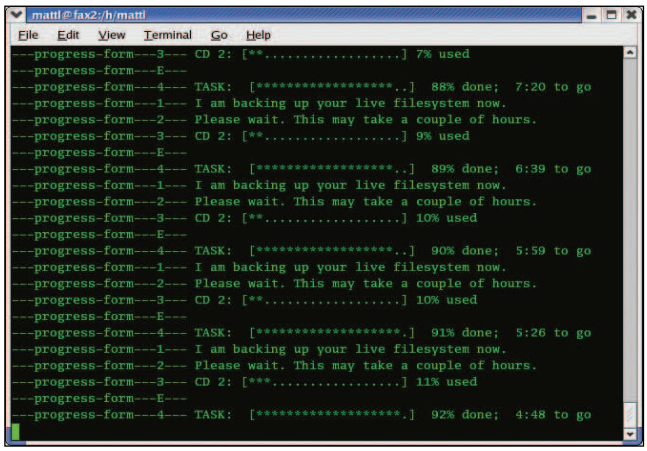
Ha ez a seregnyi felhasználási terület még mindig nem lenne elég, eláruljuk, hogy a `mondoarchive` képes összehasonlítani a mentés állományait a rendszeren pillanatnyilag lévő állományokkal. Ez az időben változatlan rendszerek épségének ellenőrzésére használható.

Láthatjuk, hogy a `mondoarchive` rugalmas és hasznos mentő és másoló segédprogram. Ha értékesek az adataink, illetve kiszolgálónk beállításai, akkor olvassunk tovább!

Függőségek

Az itt bemutatott példatelepítés a Red Hat 8.0-ra vonatkozik, ez a mi kísérleti környezetünk. Számos függőséget ki kell elégíteni a `mondoarchive` telepítése előtt. A Red Hat rendszereken a következő csomagok meglétét kell ellenőriznünk: `afio`, `cdrecord`, `buffer`, `mkisofs`, `syslinux` és `bzip2`. Ha ezek mind telepítve vannak, akkor továbbléphetünk a `mondoarchive` csomagjainak a telepítésére. Ha ezek közül valamelyik nincsen telepítve, akkor egy kis előkészítő munkára lesz szükség. Ezek a csomagok a legtöbb esetben megtalálhatók a Red Hat-korongokon. Letölthetjük őket a Red Hat webhelyéről vagy a ➲ http://www.rpmfind.net címről, de a Mondo honlapjáról is. Miután az összes csomagot összegyűjtöttük, telepítsük őket az RPM programmal.

A függőségek kielégítése után telepíthetjük a `mondoarchive` csomagjait. Két csomag kiemelten fontos: a `mindi` és a `mondo`. Ezeket beszerezhetjük a ➲ http://www.microwerks.net/~hugo/index.html oldalról.



1. kép A mondoarchive futás közben

A `mindi` a Mondónak az a része, amelyik rendszerindító és gyökér-fájlrendszert tartalmazó hajlékonylemezeket és CD-ket készít. Gyakorlatilag ez tartalmaz mindent, amire a `mondoarchive` programnak a mentéslemezek indításához szüksége van. Telepítsük először a `mindi`-t:

```
# rpm -Uvh mindi
A mindi telepítése  után telepíteni
➥kell a mondo-t is:
# rpm -Uvh mondo
```

Végrehajtás

A különféle rendszerekről sokféleképpen készíthetünk biztonsági mentést a `mondoarchive` programmal. Most csak azokat a helyzeteket ismertetjük, amelyekről már szó volt a korábbiakban, azaz a kiszolgálók biztonsági mentését és rendszerek másolatának elkészítését.
Környezetünkben számos kiszolgáló dolgozik különféle feladatokon, és mindegyik beállítása más és más. Némelyikben több IDE- vagy SCSI-merevlemez van, mert nagy mennyiségű adatot tárolnak, míg más kiszolgálók csak egy IDE- vagy SCSI-meghajtóval rendelkeznek. Akad néhány RAID rendszer is. Bizonyos kiszolgálókon állandóan változnak az adatok, máshol pedig szinte soha. A `mondoarchive` ezek mindegyikét képes lemásolni.
Jó ötlet a lemezhasználatot először kiszolgálónként megvizsgálni. Figyeljünk oda, hogy mi hova lett befűzve – nincs értelme érdektelen adatokról biztonsági mentést készíteni. Ha olyan nagyméretű könyvtáraink vannak, amelyekben nincsenek fontos adatok, hagyjuk ki őket. Nálunk például NFS és `automount` használatával több megosztott könyvár is be van fűzve minden egyes kiszolgálóra. Nem szeretnénk ezt a tényt figyelmen kívül hagyni, és a `mondoarchive` programmal ezeket az adatokat is menteni. Ha azonosítottuk a feleslegesen befűzött lemezrészeket vagy megosztásokat, a `-E` kapcsoló segítségével kizárhatjuk őket. A kapcsolót a `-E /a /b /c` alakban kell használni, ahol a `/a`, `/b` és `/c` könyvtárakat jelentenek.

Az lemezlenyomat tárolása és felírása

Most már tudjuk, hogy pontosan mit akarunk menteni, vizsgáljuk meg a `mondoarchive` parancs néhány kapcsolóját.
A biztonsági mentés készíthető CD-re, ISO-fájlba vagy NFS-megosztásra. Ebben a cikkben az lemezlenyomatok elkészítését tárgyaljuk, amelyeket egy későbbi időpontban CD-re lehet írni.
A `mondoarchive` teljes leírása megtalálható a súgóoldalán.

Mielőtt futtatnánk a `mondoarchive` parancsot, válasszunk ki a meghajtón egy sok üres helyet tartalmazó részt, hogy a nagy ISO-fájlok elférjenek. Például válasszuk a */home/mondo* könyvtárat a 6 GB-os */home* lemezrészen. A használandó parancs így néz ki:

```
# mondoarchive -Oi -d /home/mondo
➥-E "/home/mondo"
```

A `-Oi` azt jelenti, hogy a fájlrendszert lemezlenyomatok formájában kell menteni. A `-d /home/mondo` kapcsoló hatására a `mondoarchive` a */home/mondo* könyvtárba teszi az elkészült ISO-fájlokat. A rendszer méretétől függően több lemezlenyomat is létrejöhet. Végül a `-E` kapcsoló arra utasítja a programot, hogy hagyja ki a */home/mondo* könyvtárat, amelyben régi, nagyméretű lemezlenyomatok lehetnek, és feleslegesen növelnék a biztonsági mentés méretét.
Ha kevés szabad lemezterülettel rendelkezünk, egy munkakönyvtárat is meg kell adni. Ez egy ideiglenes könyvtár, amelyet a `mondoarchive` a lemezlenyomatok felépítésére használ, mielőtt véglegesen kiírná őket. Érdemes egy nagy lemezrészen elhelyezkedő munkakönyvtárat megadni, ellenkező esetben a `mondoarchive` futása sikertelen lesz, amikor elfogy a hely. Az alábbi példában feltettük, hogy a */var/local/data* egy nagy lemezrész. A munkakönyvtárat a `-S` kapcsoló után adhatjuk meg:

```
# mondoarchive -Oi -d /home/mondo
➥-S /var/local/data -E "/home/mondo"
```

A parancs elindítása után a `mondoarchive` ellenőrzi a rendszert, és ha mindent rendben talál, akkor elkezdi a biztonsági mentést. A folyamat előrehaladtáról a felhasználó visszajelzést kap (*1. kép*). Eltart egy ideig, amíg végez. A folyamat végén a program megkérdezi, hogy akarunk-e indítólemezt készíteni. Válaszolhatunk nemet, mert a felírt CD-ről lehet rendszert indítani. Ha mégis szükségünk lenne az indítólemezre, válaszoljunk igennel.
Végül a */home/mondo* könyvtárba (vagy amit megadtunk) kerülnek azok a lemezlenyomatok, ezeket lehet a korongra írni. Sokféleképpen égethetünk CD-t, például az Xcdroast, a Webmin vagy a cdrecord programok használatával. A leggyorsabb módszer első lépése a `cdrecord -scanbus` parancs, amely felderíti a CD-író adatait – sín, cél, és logikai eszközszám (LUN) –, ez általában 0,0,0. Ezeket az adatokat kell átadni a második lépésben a tényleges CD-írás elvégzéséhez:

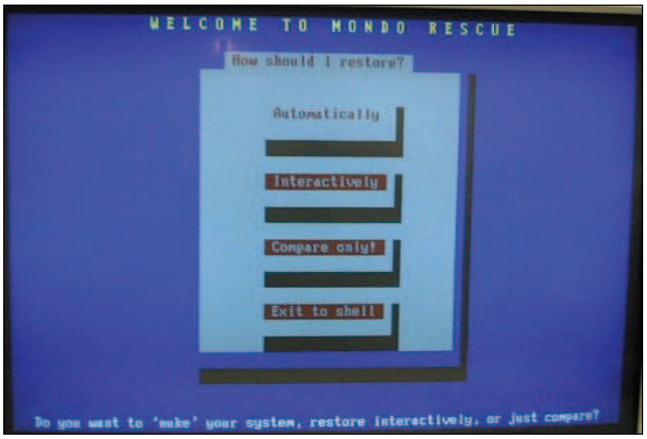
```
# cdrecord dev=0,0,0 speed=xx
➥/home/mondo/1.iso
```

A CD-írás végeredménye egy helyreállítólemez, amelyről kiszolgálóhiba esetén helyreállítható a rendszer.

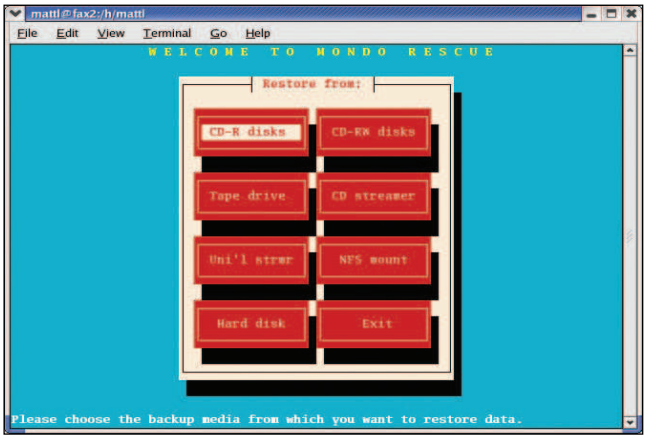
Futtatás a cron segítségével

A `mondoarchive` önműködően is futtatható az általunk megadott időben a `cron` szolgáltatás segítségével. Hozzunk létre az alábbihoz hasonló parancsfájlt, és helyezzük el a */etc/cron.daily/* könyvtárban:

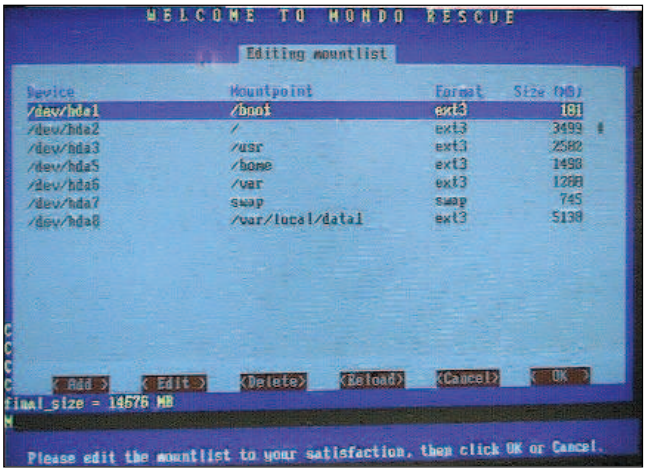
```
#!/bin/sh
mkdir -p /home/mondo/`date +%A` &&
➥mondoarchive -Oi -d /home/mondo/`date +%A`
➥-E /home/mondo
```



2. kép A Mondo párbeszédes üzemmódja



3. kép Adatforrás kiválasztása



4. kép Lemezrészadatok szerkesztése

A */etc/cron.daily/* könyvtárban elhelyezett parancsfájl minden nap ugyanabban az időpontban fut le. A végrehajtás során a */home/mondo* könyvtárba az adott napnak megfelelő könyvtárat hoz létre. Ha a hét minden napján futtatjuk a feladatot, akkor hét alkönyvtár jön létre a */home/mondo* könyvtárban, mindegyiknek a neve a hét adott napjának a neve lesz, és az aznapi biztonsági mentés lemezlenyomatai lesznek benne. Természetesen, ha korongon szeretnénk látni őket, akkor ismét a `cdrecord` parancsot kell futtatni.

Helyreállítás

Miután elkészültek a CD-k, hogyan tovább? A CD-ről elindítható a rendszer, ezért egyszerűen csak be kell tenni a meghajtóba, és máris megkezdődhet a `mondoarchive` helyreállítási képességeinek a tanulmányozása. Négy helyreállítási mód közül lehet választani: ***nuke*** (teljes), ***interactive*** (párbeszédes), ***expert*** (szakértő) és ***advanced*** (haladó).
Ha a `mondoarchive` programot kizárólag katasztrófa utáni helyreállításra használjuk, az egyetlen számunkra érdekes lehetőség a ***nuke*** mód. A lemez elindulása után írjuk be a `nuke` parancsot a parancssorba, és várjuk meg, amíg a `mondoarchive` teljesíti feladatát. A helyreállítás után a rendszer pontosan abban az állapotban lesz, mint a `mondoarchive` programmal készített mentés előtt volt. Ez jól jöhet akkor, amikor a kiszolgáló súlyos hibája miatt gyorsan helyre kell állítani a működő állapotot. Akkor is használható a módszer, ha kísérleti rendszerekkel dolgozunk, és vissza szeretnénk térni a fejlesztés egy korábbi állapotába.
A `mondoarchive` párbeszédes módja is hasznos lehet. Ebben az üzemmódban a helyreállítás előtt szerkeszthetjük a lemezrész-felosztási táblát (partition table). Ez a lehetőség akkor jön jól, ha különböző méretű merevlemezeket tartalmazó számítógépeket akarunk klónozni. A `mondoarchive` párbeszédes módjában a helyreállítás célpontjaként megjelölt lemezrészeket a programba beépített lemezrészszerkesztő segítségével át lehet méretezni. A `mondoarchive` a ***nuke*** üzemmódban is képes átméretezni a lemezrészeket, de ekkor nincs lehetőségünk a méretek megadására. A `mondoarchive` párbeszédes üzemmódjában az előre javasolt érték megváltoztatható.
A `mondoarchive` párbeszédes módjának elindításához írjuk be a parancssorba az `interactive` parancsot, miután a rendszer elindult a CD-ről. Egy menü jelenik meg, amelyből kiválaszthatjuk, hogy milyen módon kívánjuk helyreállítani az adatokat (*2. kép*); három lehetőség van: ***Automatically***, ***Interactively*** és ***Compare***. Most az ***Interactively*** menüpontot kell választanunk. Ezután a program megkérdezi, hogy honnan vegye az adatokat (*3. kép*).
Válasszuk ki a megfelelő helyet, majd lépjünk tovább. Megjelenik a lemezrészkezelő képernyő (*4. kép*), amelyen a lemezrészek adatait lehet szerkeszteni. Megváltoztathatjuk az eszköz nevét, a befűzési pontot, a fájlrendszer formátumát és a lemezrész méretét. A változtatások után nyomjuk meg az ***OK*** gombot. Ezután a Mondo néhány kérdést tesz fel a helyreállítási művelettel kapcsolatban:

- ***Are you sure you want to save your mountlist and continue***? (Biztosan mented a befűzési listát, és folytatod?) ***YES***
- ***Do you want to erase and partition your hard drives?*** (Törlöd és újra felosztod a merevlemezt?) ***YES***
- ***Do you want me to restore all your data?*** (Minden adatot helyreállítasz?) ***YES***
- ***Initialize the bootloader?*** (Alaphelyzetbe állítod a rendszerbetöltőt?) ***YES***
- ***Did you change the mountlist?*** (Megváltoztattad a befűzési listát?) ***YES***
- ***You will now edit fstab and lilo.conf/grub.conf, to make sure they match your new mountlist***. (Szerkeszd az ***fstab*** és a ***lilo.conf/grub.conf*** állományokat, hogy megegyezzenek az új befűzési listával.) ***OK***

A Mondo megnyitja a */etc/fstab* és a ***lilo.conf/grub.conf*** állományokat a `vi` szerkesztőben. A bejegyzéseket az új befűzési listának megfelelően meg kell változtatni:

- ***Edit them again?*** (Ismét szerkeszted őket?) ***NO***
- ***Label your ext2 and ext3 partitions if necessary?*** (Címkékkel látod el az `ext2`-es és `ext3`-as lemezrészeket, ha szükséges?) ***YES***

A Mondo a folyamat befejeztével a számítógép újraindítására szólít fel. Nyomjuk meg a CTRL-ALT-DELETE billentyűkombinációt.

A programkészlet ellenőrzése

A teljes helyreállítás mellett a `mondoarchive` program másra is alkalmas. Ha a lemásolt gép adatai időben változatlanok, akkor feltehető, hogy nem is akarjuk, hogy megváltozzanak. Ez a helyzet a Midwest Tool & Die linuxos útválasztóival. A `mondoarchive` programmal összehasonlíthatjuk a számítógépen lévő programkészletet a mentett adatokkal, és a változásokról jelentést kaphatunk. A folyamat egyszerű: indítsuk el a rendszert az 1. CD-ről, és írjuk be a `# compare` parancsot. A végén a `mondoarchive` az összehasonlítás eredményét a */tmp/changed.txt* állományba menti. Különbségek lesznek a CD és az első fájlrendszer között. A naplóállományok és szinte minden a */var* könyvtárban megváltozhat – ezek miatt nem kell aggódnunk. Leginkább a */etc* és a */boot* könyvtárak változásait kell figyelemmel követnünk. Ezekben a könyvtárakban olyan rendszerre jellemző beállítóállományok vannak, amelyeknek a rendszer beüzemelése után nem szabad megváltozniuk. Ha ez mégis megtörtént, és nem emlékszünk rá, hogy mi változtattuk meg őket, akkor valószínűleg nagy gondban vagyunk. Az összehasonlításnak az az egyetlen hátránya, hogy a rendszert az ellenőrzés idejére le kell állítani. Valahol ez nem gond, de akadnak olyan rendszerek, amelyeknek folyamatosan működniük kell. Ezekben az esetekben az előre tervezett leállás alkalmával kell az ellenőrzést elvégezni. A másik lehetőség a rendszer ellenőrzésére a népszerű Tripwire alkalmazás használata. Inkább a Tripwire programot javasoljuk az olyan időben állandó rendszerek ellenőrzésére, amelyeknél fontos a folyamatos üzem.

Összegzés

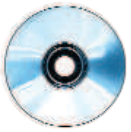
Megismertük a `mondoarchive` képességeit a katasztrófa utáni helyreállítás terén. A program részletesebb felfedezését javasoljuk – segítségével a katasztrófa után gyorsan és teljesen helyreállíthatjuk a rendszereinket. Mindehhez látogassunk el a Mondo Rescue honlapjára (➲ http://www.microwerks.net/~hugo/index.html).

Linux Journal 2003. október, 114. szám

Craig Swanson (craig.swanson@slssolutions.net) Hálózattervező és Linux-tanácsadó az SLS Solutions cégnél. Emellett linuxos programokat fejleszt a Midwest Tool & Die számára. Craig 1993 óta használ Linuxot.

Matt Lung (matt.lung@slssolutions.net) Hálózatmérnökként dolgozik a Midwest Tool & Die cégnél. Ő is Linux-tanácsadó az SLS Solutionsnél.

Eszközosztályok



Árnyaljuk egy kicsit a képet, avagy hogyan írhatunk olyan eszközillesztőt, amelyik hibátlanul működik a 2.6-os rendszermag alatt.

A legutóbbi, „A rendszermag beállításai” című írásomban (Linuxvilág, 2003 júniusi szám) megismerkedtünk a rendszermag illesztőprogram-keretrendszerével, illetve az általános busz és az illesztőprogramok és -eszközök kódjának a működésével. A különböző alrendszerek működésének leírásakor az i2c magot hoztam fel példaként. Ebben a hónapban az illesztőprogram-osztálykód működése kerül terítékre, és ismét az i2c kódja szolgál példaként.
Mint a múlt alkalommal már szó esett róla, az eszközosztályok nem az objektumközpontú világ értelmezése szerinti osztályokat jelentik, inkább adott típusú szolgáltatást nyújtanak a felhasználó számára. A rendszermag osztályai például a tty, a blokkos vagy a hálózati eszközök és – hamarosan – a fájlrendszerek.
A 2.5.69-es rendszermagban teljesen újraírt illesztőprogram-osztályrendszer mutatkozott be. A korábbi rendszermagváltozatokban az osztályok támogatása szoros kapcsolatban állt az illesztőprogramok és az eszközök támogatásával. Az adott osztály egyben az eszközhöz is kötődött, amikor egy illesztőprogramnál bejegyzésre került. Ez sok eszköznél és osztálynál jól működött, de a valós élet bőségesen szolgált olyan eszközökkel, amelyek nem nagyon illettek ebbe a modellbe. Most az osztályok támogatása csak laza kapcsolatban áll az illesztőprogramokkal és az eszközökkel, eszközre vagy illesztőprogramra valójában már nincs is szükség az osztálykód használatához, mint ezt a tty-osztálykód is mutatja. Az osztálykód három különböző típusú szerkezetre osztható: osztályokra, osztályeszközökre és osztályfelületekre.

Osztályok

A rendszermagban az osztályok megadása egy `struct class` adatszerkezettel történik. Igen, C-ben a class nem fenntartott szó. (Mindenki, aki C++ fordítóval szeretne rendszermagot fordítani, az új osztálykód készítőjéhez címezze jókívánságait.) Ha létre akarunk hozni egy osztály-adatszerkezetet, mindössze a `struct class` adatszerkezet `name` változójának kell értéket adnunk, és máris érvényes osztályt kaptunk. Ezt az alábbi kóddal tehetjük meg:

```
static struct class i2c_adapter_class = {
       .name = "i2c_adapter"
};
```

Az osztály adatszerkezetet megadás után a `class_register` függvény meghívásával lehet bejegyezni az illesztőprogram-magnál:

```
if (class_register(&i2c_adapter_class) != 0)
    printk(KERN_ERR "i2c adapter class
           ➥failed "
           "to register properly\n");
```

Ha a `class_register` függvény hibajelzés nélkül tér vissza, akkor a */sys/class/i2c_adapter* könyvtár sikeresen létrejött.

Később, ha az osztályt el kell távolítani, a `class_unregister` függvényt kell meghívni:

```
class_unregister(&i2c_adapter_class);
```

Osztályeszközök

Az osztályok a különféle osztályeszközök kezelésére szolgálnak. A rendszermagban az osztályeszközök megadása a `struct class_device` adatszerkezettel történik. Ez az adatszerkezet számos, az illesztőprogram-mag által használt változót tartalmaz, ezekkel az illesztőprogram írójának nem kell foglalkoznia. Az általa értékkel ellátandó változók a következők:

- `class`: arra a `struct class` adatszerkezetre kell mutatnia, amelynek a feladata lesz az osztályeszköz kezelése.
- `dev`: az osztályeszközhöz hozzárendelt `struct device` adatszerkezet címét kell értékéül adni, amennyiben van ilyen. Egyetlen `struct device` adatszerkezetre több osztályeszköz-adatszerkezet is hivatkozhat. Ez a legfőbb különbség az előző rendszermag osztálytámogatása és a jelenlegi megvalósítás között. Ennek a változónak nem muszáj értéket adni, a rendszermag nélküle is jól működik. Ha mégis kap értéket, egy a `struct device` adatszerkezetre mutató közvetett eszközhivatkozás jön létre az osztályeszköz `sysfs` bejegyzésében. A példát lásd lejjebb.
- `class_id`: karaktertömb, az osztályeszköz leírását tartalmazza. Az adott osztály-adatszerkezethez hozzárendelt osztályeszköz-adatszerkezetek között egyedinek kell lennie.
- `class_data`: egy mutatót tartalmaz, amely az osztály-illesztőprogram által az osztályeszközhöz hozzárendelendő saját adatokra mutat. Ezt a változót közvetlenül nem kell elérni, értékének beállítására és lekérdezésére a `class_set_devdata` és a `class_get_devdata` függvény használható.

Megfelelően előkészített struct `class_device` adatszerkezet bejegyzését a `class_device_register` függvénnyel lehet elvégezni. Egy `struct class_device` adatszerkezet kezdeti értékadására és az illesztőprogram-magnál való bejegyzésére az alábbi, a ***drivers/i2c/i2c-core.c*** fájlból származó kódrészlet ad példát:

```
/* A csatoló hozzáadása az i2c_adapter
osztályhoz */

memset(&adap->class_dev, 0x00,
       sizeof(struct class_device));
adap->class_dev.dev = &adap->dev;
adap->class_dev.class = &i2c_adapter_class;
strncpy(adap->class_dev.class_id,
        adap->dev.bus_id, BUS_ID_SIZE);
class_device_register(&adap->class_dev);
```

Először a `struct class_device` változó (a `struct`

i2c_adapter változóba van beágyazva) kap kezdeti értékként nullát. Bejegyzés előtt minden illesztőprogrammodell-adatszerkezet összes változójának nullát kell értékként adni, az illesztőprogram-mag csak így tudja helyesen kezelni őket. Ezután a dev változó olyan értéket kap, hogy az i2c_adapter struct device változójára mutasson. Ebben az esetben a struct i2c_adapter adatszerkezet egy struct device és egy struct class_device adatszerkezetet is tartalmaz. A class változó az i2c_adapter_class változó címére fog mutatni, majd a class_id változó az eszköz bus_id azonosítójával megegyező értéket kap. Mivel az i2c_adapter eszköz bus_id azonosítója egyedi értékkel bír, egyben az i2c_adapter class_device adatszerkezetének class_id összetevője is egyedi lesz. Végül a class_device_register függvény meghívásával megtörténik az osztályeszköz-adatszerkezet bejegyzése a rendszermag eszközillesztő magjánál.
A fenti kód és két i2c csatoló betöltése után a próbagépen az *1. litstán* láthatóak szerint alakult a */sys/class/i2c_adapter* fa.

A fa alapján is látható, hogy az illesztőprogram-mag önműködően hoz létre egy eszköz- és egy illesztőprogram közvetett hivatkozást, és ezek a sysfs fa megfelelő pontjaira mutatnak. Ha a dev mutató nem a struct device adatszerkezetre mutat, akkor a közvetett hivatkozások nem jönnek létre. Ha bepillantunk a */sys/class/tty* könyvtárba, akkor az osztályeszköz-bejegyzések túlnyomó részéhez nem tartozik struct device adatszerkezet, vagyis ezek a szimbolikus hivatkozások hiányoznak.

Osztályfelületek

Az osztályfelületek révén tudja a rendszer értesíteni a saját kódunkat, ha egy struct class_device adatszerkezet egy adott osztálynál bejegyzésre, illetve eltávolításra kerül. Az osztályfelület megadása a struct class_interface adatszerkezettel történik, ami egészen egyszerű:

```
struct class_interface {
    struct list_head  node;
    struct class      *class;
    int (*add)        (struct class_device *);
    void (*remove)    (struct class_device *);
};*
```

A class változó értékeként kell megadni azt az osztályt, amelynek az eseményeiről értesülni szeretnénk. Az add (hozzáadás) és a remove (eltávolítás) változó értéke egy függvény lesz, amelyet akkor kell meghívni, ha eszközt adunk hozzá az osztályhoz, illetve távolítunk el belőle. Ha valamelyik eseményről nem akarunk értesülni, akkor a neki megfelelő változónak nem kötelező értéket adnunk.
Ha osztályfelületet szeretnénk bejegyezni a rendszermagnál, a class_interface_register függvényt kell használnunk, míg az eltávolítást a class_interface_unregister függvénnyel hajthatjuk végre. Az osztályfelületek használatára szép példát találunk a processzor órajel-magkódjában, amely a rendszermag forrásának ***kernel/cpufreq.c*** fájljában található meg.

Fájlok létrehozása

Mint már említettem, az i2c-adapter osztály segítségével könnyen felismerhetjük a rendszerben lévő különféle i2c-csatolókat, illetve meghatározhatjuk helyüket az illesztőprogram-fában. Csakhogy az i2c-csatolókat a felhasználó nem címezheti meg közvetlenül. Amennyiben párbeszédbe szeretnénk elegyedni egy i2c-csatolóval, akkor be kell töltenünk egy i2c-lapka

```
$ tree /sys/class/i2c-adapter/
/sys/class/i2c-adapter/
|-- i2c-0
|   |-- device
    ➥-> ../../../devices/pci0/00:07.3/i2c-0
|   `-- driver ->
../../../bus/i2c/drivers/i2c_adapter
|-- i2c-2
    |-- device
    ➥-> ../../../devices/legacy/i2c-2
    `-- driver
    ➥-> ../../../bus/i2c/drivers/i2c_adapter
```

illesztőprogramot, vagy az i2c-dev illesztőprogramot kell használnunk. Az i2c-dev illesztőprogram karakteres illesztőfelületet biztosít a rendszerben jelenlévő összes i2c-csatolóhoz. Mivel nem árt pontosan tudni, hogy mely i2c-dev eszközök mely i2c-csatolókhoz csatlakoznak, létrehoztuk az i2c-dev osztályt:

```
static struct class i2c_dev_class = {
    .name    = "i2c-dev"
};
```

Miután az i2c-dev illesztőprogram az összes i2c-csatolót megtalálta, az eszközillesztő mag egy új i2c osztályeszközzel bővül. A hozzáadás az i2c_add_class_device függvényen belül történik:

```
static void
i2c_add_class_device(char *name, int minor,
                     struct i2c_adapter *adap)
{
    struct i2c_dev *i2c_dev;
    int retval;

    i2c_dev = kmalloc(sizeof(*i2c_dev),
GFP_KERNEL);
    if (!i2c_dev)
        return;
    memset(i2c_dev, 0x00, sizeof(*i2c_dev));

    if (adap->dev.parent == &legacy_bus)
        i2c_dev->class_dev.dev = &adap->dev;
    else
        i2c_dev->class_dev.dev =
        ➥adap->dev.parent;
    i2c_dev->class_dev.class = &i2c_dev_class;
    snprintf(i2c_dev->class_dev.class_id,
             BUS_ID_SIZE, "%s", name);
    retval =
        class_device_register
        ➥(&i2c_dev->class_dev);
    if (retval)
        goto error;
    class_device_create_file
    ➥(&i2c_dev->class_dev,
    ➥&class_device_attr_dev);

    i2c_dev->minor = minor;
    spin_lock(&i2c_dev_list_lock);
    list_add(&i2c_dev->node, &i2c_dev_list);
```

```
$ tree /sys/class/i2c-dev/
/sys/class/i2c-dev/
|-- i2c-0
|    |-- dev
|    |-- device
     ➥-> ../../../devices/pci0/00:07.3
|    `-- driver ->
../../../bus/pci/drivers/piix4-smbus
`-- i2c-2
     |-- dev
     |-- device
     ➥-> ../../../devices/legacy/i2c-2
     `-- driver
     ➥-> ../../../bus/i2c/drivers/i2c_adapter
```

```
    spin_unlock(&i2c_dev_list_lock);
    return;
error:
    kfree(i2c_dev);
}
```

A függvény csak néhány részletben tér el az `i2c_adapter` osztály bejegyzését végző kódtól. Először is a ***class_dev.dev*** mező vagy a csatoló szülőeszközére vagy magára a csatolóra mutat. Ennek oka az, hogy az i2c-csatolóknak valójában nincs szülőjük az általános rendszermag-eszközfában, hiszen vagy a rendszermag illesztőprogram-modelljében (például az ISA busszal ellentétben) nem szereplő buszon találhatók, vagy lényegében nem is csatlakoznak semmilyen buszra (mint például bizonyos beágyazott i2c-vezérlők). Ha egy i2c-csatolónak nincs helye a rendszermag eszközfájában, a rendszer az örökölt buszhoz rendeli hozzá. Az örökölt busz – a */**sys**/**devices**/**legacy*** alatt található – hivatott az összes ilyen eszköz kezelésére.
A második eltérést ennél az osztályeszköznél az alábbi sorban fedezhetjük fel:

```
class_device_create_file (&i2c_dev->class_dev,
➥&class_device_attr_dev);
```

A `class_device_create_file` függvény egy fájlt hoz létre az osztályeszköz könyvtárában. A fájlnevet és a jellemzőket a `CLASS_DEVICE_ATTR` makró adja meg az alábbiak szerint:

```
static ssize_t
show_dev(struct class_device *class_dev,
         ➥char *buf)
{
    struct i2c_dev *i2c_dev =
to_i2c_dev(class_dev);
    return sprintf(buf, "%04x\n",
           ➥MKDEV(I2C_MAJOR, i2c_dev->minor));
}
static
CLASS_DEVICE_ATTR(dev, S_IRUGO, show_dev,
                  ➥NULL);
```

A `CLASS_DEVICE_ATTR*` makró megadása a következő:

```
#define
CLASS_DEVICE_ATTR(_name,_mode,_show,_store)
```

```
$ tree /sys/bus/i2c/
/sys/bus/i2c/
|-- devices
|   |-- 0-0050 ->
../../../devices/pci0/00:07.3/i2c-0/0-0050
|   |-- 0-0051 ->
➥../../../devices/pci0/00:07.3/i2c-0/0-0051
|   |-- 0-0052 ->
➥../../../devices/pci0/00:07.3/i2c-0/0-0052
|   |-- 0-0053 ->
➥../../../devices/pci0/00:07.3/i2c-0/0-0053
|   `-- 2-0290 -> ../../../devices/legacy/
➥i2c-2/2-0290
`-- drivers
    |-- dev driver
    |-- eeprom
    |    |-- 0-0050 ->
➥../../../../devices/pci0/00:07.3/i2c-0/0-0050
    |    |-- 0-0051 ->
➥../../../../devices/pci0/00:07.3/i2c-0/0-0051
    |    |-- 0-0052 ->
➥../../../../devices/pci0/00:07.3/i2c-0/0-0052
    |    `-- 0-0053 ->
➥../../../../devices/pci0/00:07.3/i2c-0/0-0053
    |-- i2c_adapter
    `-- w83781d
         `-- 2-0290 ->
➥../../../../devices/legacy/i2c-2/2-0290
```

Az i2c-csatolókhoz tartozó */**sys**/**devices**/* könyvtárak tartalma:

```
$ tree /sys/devices/pci0/00:07.3
/sys/devices/pci0/00:07.3
|-- class
|-- device
|-- i2c-0
|    |-- 0-0050
|    |    |-- eeprom_00
|    |    |-- name
|    |    `-- power
|    |-- 0-0051
|    |    |-- eeprom_00
|    |    |-- name
|    |    `-- power
|    |-- 0-0052
|    |    |-- eeprom_00
|    |    |-- name
|    |    `-- power
|    |-- 0-0053
|    |    |-- eeprom_00
|    |    |-- name
|    |    `-- power
|    |-- name
|    `-- power
|-- irq
|-- name
|-- power
|-- resource
|-- subsystem_device
|-- subsystem_vendor
`-- vendor*
```

```
struct class_device_attribute
class_device_attr_##_name = {
     .attr  = {.name = __stringify(_name),
               .mode = _mode },
     .show  = _show,
     .store = _store,
};
```

A CLASS_DEVICE_ATTR* makrón belül az alábbi átadott értékeket találjuk:

- _name: a sysfs-en belül létrehozandó fájl neve, illetve része a name változónak, amely ezt a teljes jellemzőt írja le.
- _mode: a fájlhozzáférési mód, amellyel a fájl létrehozása történik. A megfelelő érték megadásához a szabványos hozzáférési makrók használhatók.
- _show: arra a függvényre mutat, amelynek meghívására a fájl tartalmának olvasásakor kerül sor. Ennek a függvénynek az alábbi visszatérési értékkel és átadott értékekkel kell rendelkeznie. A változónak nem kell értéket adni, ha a fájlból nem fogunk olvasni.

```
ssize_t
show (struct class_device *class_dev,
      ➥char *buf);
```

- _store: arra a függvényre mutat, amelynek a meghívására a fájlba való íráskor kerül sor. Ennek a függvénynek az alábbi visszatérési értékkel és átadott értékekkel kell rendelkeznie. A változónak nem kell értéket adni, ha a fájlba nem fogunk írni.

```
ssize_t
store (struct device *dev,
       ➥const char *buf, size_t count);*
```

Szinte minden illesztőprogram-modell adatszerkezete rendelkezik egy ATTR() makróval, amely megad egy fájlt a sysfs fán belül.
Ebben a példában egy ***dev*** nevű fájl jön létre a class_device_create_file függvény meghívásakor. A fájlból minden felhasználó csak olvashat. A fájl tartalmának olvasásakor az illesztőprogrammag a show_dev függvényt hívja meg. A show_dev függvény feltölti a neki átadott átmeneti tárat a felhasználónak továbbítandó adatokkal. Ebben az esetben a felhasználó az adott eszköz fő- és alszámát kapja meg. Minden fő- és alszámot használó osztályeszköznek rendelkeznie kell egy ***dev*** fájllal a megfelelő ***sysfs*** osztályeszköz könyvtáron belül.
A class_device_create_file függvény által létrehozott fájlok eltávolítására a class_device_remove_file függvény használható. Az eszközök eltávolításakor semmilyen fájlt nem kell kézzel törölni. Amikor egy eszközt eltávolítunk a sysfs alól, akkor a neki megfelelő könyvtárból a sysfs-mag önműködően törli az összes fájlt. Ha tehát eltávolítjuk az i2c-dev osztályeszközt a rendszerből, mindössze az alábbi parancsokra van szükség:

```
static void
i2c_remove_class_device(int minor)
{
     struct i2c_dev *i2c_dev = NULL;
     struct list_head *tmp;
     int found = 0;

     spin_lock(&i2c_dev_list_lock);
     list_for_each (tmp, &i2c_dev_list) {
          i2c_dev = list_entry(tmp, struct
                    ➥i2c_dev, node);
          if (i2c_dev->minor == minor) {
               found = 1;
               break;
          }
     }
     if (found) {
          list_del(&i2c_dev->node);
          spin_unlock(&i2c_dev_list_lock);
          class_device_unregister
                    ➥(&i2c_dev->class_dev);
          kfree(i2c_dev);
     } else {
     spin_unlock(&i2c_dev_list_lock);
     }
}
```

Áttekintés

Ha az i2c-dev illesztőprogram és két i2c-csatoló illesztőprogram (az i2c-piix4 és az i2c-isa) be van töltve, akkor a */**sys**/**class**/**i2c-dev*** könyvtár tartalma a *2. listában* láthatók szerint alakul.

A */**sys**/**class**/**i2c-dev**/**i2c-2**/* könyvtárban lévő ***dev*** fájl tartalma az alábbi karakterlánc lesz:

```
$ cat /sys/class/i2c-dev/i2c-2/dev 5902
```

Ez a 86-os fő- és a 2-es alszámhoz tartozik, ezek az adott eszköz karakteres fő- és alszámai.
A */**sys**/**bus**/**i2c**/* könyvtár tartalma az alábbiak szerint alakul, miután néhány i2c ügyfél-illesztőprogramot betöltöttünk (*3. lista*). A 4. listát lásd az 53. CD Magazin/Eszköz könyvtárában.
A rendszermag illesztőprogram-modellje által alkalmazott, egymáshoz kapcsolódó adatszerkezet-mutatók rendszeréről, illetve felhasználói alkalmazásukról ***Jonathan Corbet*** készítette a legjobb leírást: „Web woven by a spider on drugs" (➲ http://lwn.net/Articles/31185/). Bízom abban, hogy két írásom alapján kicsit áttekinthetőbbé vált ez a hatalmas kuszaság, és sikerült valamennyire megismerkedni az eszközök között a rendszermagon belül fennálló kapcsolatokkal.

Köszönetnyilvánítás

Szeretnék köszönetet mondani ***Pat Mochel***-nek, amiért ilyen kiváló és átfogó, a felhasználók számára is szemléletes keretrendszert készített az összes rendszermag-illesztőprogramhoz és eszközhöz. Ugyancsak hálával tartozunk mindenkinek, aki valamelyik rendszermagillesztőprogram-alrendszert tartja karban, és örömmel alakította át alrendszerét az új modellnek megfelelően. Segítségük nélkül az eszközillesztő kódja nem lett volna több tetszetős egyetemi feladványnál.

Linux Journal 2003. augusztus, 112. szám

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

Hitelesítés LDAP használatával (3. rész)

Az LDAP-ról szóló sorozat záró írásában LDAP-kiszolgálónk Mick segítségével végre eljut odáig, hogy valódi felhasználókat valóságos környezetben is képes hitelesíteni.

Az elmúlt hónapokban egy LDAP-kiszolgálót építgettünk. Telepítettük az OpenLDAP-ot; beállítottuk a `slapd`-t, a kiszolgálódémont; életet leheltünk a TLS alapú titkosításba; végül létrehoztuk első LDAP-bejegyzésünket, egy fő szervezet bejegyzését. Elérkezett az idő, hogy a rendszerhez hozzáadjuk a felhasználókat, és IMAP-munkamenetek hitelesítésére használjuk a kiszolgálót.

Az adatbázis szerkezete

Az LDAP felhasználói adatbázisának létrehozásakor az első lépés a címtár szerkezetének kigondolása, amely annak eldöntését is magába foglalja, hogy a felhasználókat és az egyéb egyedeket kívánjuk-e csoportosítani, vagy egyszintű szerkezetet akarunk használni. Ha az LDAP-adatbázist kizárólag – és csakis kizárólag – hálózati telefonkönyv vagy hitelesítő kiszolgáló üzemeltetésére használjuk, akkor egy egyszintű adatbázis is megfelel. Ebben az esetben a felhasználók megkülönböztető nevei (Distinguished Name, DN) valahogy így alakulnak:

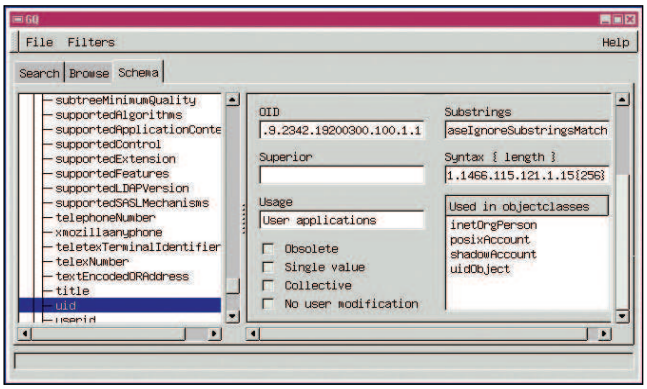
```
dn=Mick Bauer,dc=proba,dc=org
```

Ha azonban az adatbázis nemcsak személyekkel, de szervezeti csoportokkal, részlegekkel, számítógépekkel stb. kapcsolatos adatokat is tartalmaz, akkor kifinomultabb szerkezetű címtárra lehet szükség. Ilyet több módszerrel is lehet építeni. Az egyik lehetőség az, hogy a `domainComponent` (`dc`) mezőkkel tartományunkon belül altartományokat hozunk létre, függetlenül attól, hogy ezek a DNS-ben valóban léteznek-e. Ekkor a bejegyzések `dn=Bick Mauer,dc=tervezes,dc=proba,dc=org` formátumúak lesznek. Egy másik lehetőség az `organizationalUnit` objektumok hasonló módon való használata, ilyenkor `dn=Dick Lauer,ou=tervezes,dc=proba,dc=org` lesz a kapott formátum.

Annak érdekében, hogy a további mondandóm is követhető legyen, maradok az egyszintű adatbázis használatánál; természetesen senkinek nem szeretném a kedvét szegni: mindenki maga határozza meg, hogy igényeinek milyen szerkezetű LDAP-adatbázis felel meg a leginkább. A ➲ http://www.openldap.org címen elérhető és az OpenLDAP programhoz mellékelt leírásokban bőven találni példákat.

A séma és a felhasználórekordok

Szintén fontos döntés a használni kívánt és a rekordokba beillesztendő LDAP-jellemzők körének a meghatározása. A múlt alkalommal már szóltam arról, hogy ezek hogyan vannak csoportosítva a sémákban, illetve milyen kapcsolatban állnak egymással. Bizonyára mindenki emlékszik, hogy a */etc/openldap/slapd.conf* fájlban megadott sémák határozzák meg, hogy a rekordokban milyen jellemzőket lehet tárolni. A sémákat nem elég a */etc/openldap/slapd.conf* fájlban felsorolni: minden létrehozott rekordban a megfelelő sémákat `objectClass` állításokkal hozzá kell rendelni az egyes



A séma böngészése a gq segítségével

felhasználókhoz. Ugyancsak a múlt alkalommal mondtam el, hogy a */etc/openldap/schema* könyvtárban lévő sémafájlok határozzák meg, hogy milyen séma milyen jellemzőket tartalmaz, illetve hogy az adott sémán belül milyen objektumosztályokhoz milyen jellemzők tartoznak.

Tegyük fel, hogy az LDAP-kiszolgáló IMAP-kapcsolatokhoz fog hitelesítést végezni. Ekkor az `uid` és a `userPassword` LDAP-jellemzőkre mindenképpen szükség lesz. Hasonlók mondhatók el bármely más alkalmazásról is, ha az a Bind-eljárással végez LDAP alapú hitelesítést, vagyis hitelesítő programrésze egyszerűen megpróbál kapcsolódni az LDAP-kiszolgálóhoz úgy, hogy közben a felhasználó által megadott nevet és jelszót alkalmazza. Ha a csatlakozás sikeres, a program a hitelesítést sikeresnek ítéli, és lezárja az LDAP-kapcsolatot.

Azt, hogy az `uid` és a `userPassword` jellemzők mely sémában és objektumosztályokban találhatók meg, a legegyszerűbben úgy mérhetjük fel, hogy a `grep` paranccsal a */etc/openldap/schema* könyvtárban rákeresünk az `uid` és a `userPassword` karakterláncra, feljegyezzük a találatokat, kézzel végignézzük a megfelelő állományokat, és kikeressük az említett két jellemzőt `MUST()` vagy `MAY()` állításban tartalmazó objektumosztályokat. Jómagam az `uid` karakterláncra az OpenLDAP 2.0-t futtató Red Hat 7.3 rendszeren a következő fájlokban találtam hivatkozást: ***core.schema, cosine.schema, inetorgperson.schema, nis.schema*** és ***openldap.schema***.

A talált fájlokon a `less` segítségével végigszaladva a következőkre jutottam: a ***core.schema*** fájl `uidObject` objektuma megköveteli az `uid` jellemzőt, a ***cosine.schema*** egyetlen, az `uid` jellemzőre vonatkozó hivatkozása megjegyzésbe van téve, így érdektelen; az ***inetorgperson.schema*** egy `inetOrgPerson` objektumosztályt tartalmaz, amely kiegészítő jellemzőként támogatja az `uid` használatát, a ***nis.schema*** két objektumosztállyal rendelkezik – `posixAccount` és `shadowAccount` –, mindkettő megköveteli az `uid` elérhetőségét, nem különben az ***openldap.schema*** `OpenLDAPperson` objektumosztálya.

Szerencsére ezeket az adatokat sokkal gyorsabban is beszerezhetjük. A `gq` LDAP-eszköz segítségével LDAP-kiszolgálónk

1. lista LDIF fájl egy felhasználói rekord hozzáadásához

```
dn: cn=Wong Fei Hung,dc=proba,dc=org
cn: Wong Fei Hung
sn: Wong
givenname: Fei Hung
objectclass: person
objectclass: top
objectclass: inetOrgPerson
mail: wongfh@proba.org
telephonenumber: 651-344-1043
o: Proba
uid: wongfh
```

összes sémájának összes jellemzőjét végigböngészhetjük. *Képünkön* látható, hogy a saját LDAP-kiszolgálómon a `gq` szerint hol találhatók hivatkozások az `uid` jellemzőre. Képünkön látható Used in objectclasses (Ezekben az objektumosztályokban használva) lista elárulja nekünk, hogy a kiválasztott jellemző – jelen esetben az `uid` – az `uidObject`, a `posixAccount`, a `shadowAccount` és az `inetOrgPerson` objektumosztályokban tűnik fel, pontosan azokban, amelyeket a `grep` használatával is megtaláltunk. Az `OpenLDAPperson` objektumosztály nem tűnik fel a `gq` ablakában, mivel a kérdéses LDAP-kiszolgálón a */etc/openldap/slapd.conf* fájlban nincs olyan parancs, amely az ***openldap.schema*** fájl használatát előírná. Miután eldöntöttük, hogy milyen formátumú LDAP-rekordokat szeretnénk használni, bármikor eltávolíthatjuk a számunkra szükségtelen objektumosztályokat tartalmazó sémákat. Gondolom, mindez eléggé zűrös dolognak tűnik, és valóban az tud lenni, de túl kell esni rajta, hiszen olyan rekordokat kell létrehoznunk, amelyek képesek a környezetünk által igényelt adatok tárolására. Mivel az LDAP meglehetősen rugalmas, a jellemzők igényeinknek megfelelő összeválogatásához bizony szükség lehet egy kis bütykölésre.

Rekordok létrehozása és hozzáadása

Ahogy a sémák böngészése, úgy az LDAP-rekordok hozzáadása is végezhető kézzel és grafikus felületen egyaránt.
A múlt alkalommal kézzel hoztuk létre a fő szervezet bejegyzését, így első felhasználói rekordunkat is így fogjuk megalkotni. Az eljárás két lépésből áll: először készítenünk kell egy különleges, LDIF formátumú szövegfájlt, majd az `ldapadd` paranccsal be kell emelnünk a tartalmát az LDAP-adatbázisba. Vessünk egy pillantást az *1. listában* látható LDIF-fájlra.
Mivel minden mást ezek határoznak meg, az 1. lista elemzését a benne található `objectclass` állításokkal fogjuk kezdeni. Ez a felhasználó a `top` (minden rekord esetében kötelező), a `person` és az `inetorgperson` objektumosztályokkal van összerendelve. A `person` osztályt azért választottam ki, mert benne található meg a `userPassword` (ezt az 1. kódrészletben nem adtuk meg, ám Mr. Wong hamarosan jelszót is kap) és a `telephonenumber` jellemző – az utóbbira ugyan egyelőre nincs szükségünk, de a későbbiekben még lehet. Az `inetOrgPerson` objektumosztályban található az `uid` jellemző, valamint jó pár egyéb jellemző is, amelyek később még jól jöhetnek.
A sémában található `MUST` és `MAY` megszorításokat – többek között – úgy kerülhetjük meg, hogy a `schemacheck off` utasítással bővítjük a */etc/openldap/slapd.conf* fájlt. Így a ***slapd.conf*** fájlban hivatkozott sémafájlok bármelyikének bármely jellemzőjét használhatjuk, figyelmen kívül hagyva az objektumosztályokat. Természetesen ezzel rontjuk a más LDAP-kiszolgálókkal való együttműködés esélyét, sőt bizonyos alkalmazások működését meg is akadályozhatjuk, arról nem is szólva, hogy megcsúfoljuk az LDAP RFC-ket. Nem csoda, hogy a szakértők általában elutasítják a sémaellenőrzés kikapcsolásának ezt a módját.
Talán nincs szükség arra, hogy az 1. kódrészlet minden sorát pontosan elmagyarázzam, a jellemzők jelentése könnyen megérthető. Annyit azért megemlítenék, hogy bár nem kell az összes valaha használni kívánt jellemzőnek értéket adni, némelyik beállítása mégis kötelező – ezeket a megfelelő objektumosztályok megadásában `MUST()`, `MUST()` állításokban találjuk meg. Minden általunk megadott jellemzőnek legalább egy, a rekordban megadott objektumosztály `MUST()` vagy `MAY()` állításában szerepelnie kell, bizonyos jellemzők – mint a `cn` – pedig egy-egy rekordon belül többször is megadhatók.
Az 1. listában látható rekordot az `ldapadd` paranccsal adhatjuk hozzá az adatbázishoz:

```
$ ldapadd -x -D "cn=ldapproba,dc=proba,
➥dc=org" -W -f ./wong.ldif
```

Az előző alkalommal is hasonló módon alkalmaztuk az `ldapadd` parancsot. A parancs használatának módjáról az ldapadd(1) súgóoldalon találni részletes leírást.
Ha az LDIF fájlban megadott objektumosztályok által megkövetelt jellemzők mindegyikét megadtuk, ha a kiválasztott jellemzőket kivétel nélkül támogatják ezek az objektumosztályok, és ha az erre utaló parancs megjelenésekor a helyes LDAP `bind` jelszót adjuk meg, akkor a rekord bekerül az adatbázisba. Ha valamelyik feltétel nem teljesül, akkor a művelet végrehajtása sikertelen lesz, az `ldapadd` pedig kiírja, hogy milyen hiba merült fel. Vagyis: hibáinkból tanulva, próbálgatással is összeüthetjük a megfelelő formátumú rekordot. Ha egyszer már sikerült eredményre jutni, a többi rekordnál is ugyanazt a formátumot tudjuk használni, nem kell újra megszenvednünk a sémával.
Egy apróságra azért felhívnám a figyelmet. Ha az LDIF fájlunk több rekordot is tartalmaz, ami megengedett, és az LDAP-kiszolgáló hibát észlel, akkor abbahagyja a fájl feldolgozását, és a hibás után következő rekordok beolvasását meg sem kísérli. Az első néhány felhasználó hozzáadásakor tehát, amíg a rekordformátum valóban véglegessé nem válik, inkább csak egy-egy rekordot tartalmazó LDIF állományokkal érdemes dolgozni.
A rekordok kézi létrehozása kicsit unalmas, de legalább módot ad egy kis elemzésre és ismerkedésre. Ez a lehetőség különösen az LDAP-adatbázis felépítésének kezdetekor fontos.
Ha már van néhány felhasználórekordunk, akkor valamilyen grafikus felületet nyújtó eszköz, például az LDAP Browser/Editor (➲ http://www.iit.edu/~gawojar/ldap) vagy a `gq` (a legtöbb Linux-terjesztésben megtalálható) segítségével is folytathatjuk munkánkat. Ha például a `gq`-ban rákattintunk valamelyik rekordra, akkor egy menü bukkan fel, amely egy ***New Use current entry*** (Új Pillanatnyi bejegyzés használata) parancsot is tartalmaz, ezzel a kijelölt rekordból készíthetünk másolatot. Így nyilván gyorsabban és egyszerűbben dolgozhatunk, mintha mindent begépelnénk egy LDIF fájlba.

Jelszavak létrehozása

Az 1. kódrészlet kapcsán már utaltam arra, hogy az LDIF fájlokban általában nem adjuk meg a felhasználók jelszavait. Erre a célra egy külön eljárás használható, amely az `ldappasswd` parancsban „testesül meg”. Használata alapvetően az `ldapadd` parancséhoz hasonló:

```
$ ldappasswd -S -x -D
➥"cn=rendszergazda,dc=upstreamsolutions,
➥dc=com" -W "cn=Phil
➥Lesh,dc=upstreamsolutions,dc=com"
```

Az `ldappasswd` parancs használatához nem kell a héjmunkamenetbe bejelentkeznünk az LDAP-kiszolgálón, elég, ha a `-H` kapcsolóval megadjuk a távoli LDAP-kiszolgáló URL-jét:

```
$ ldappasswd -S -x -H
➥ldaps://ldap.upstreamsolutions.com -D
➥"cn=rendszergazda,dc=upstreamsolutions,
➥dc=com" -W "cn=Phil
➥Lesh,dc=upstreamsolutions,dc=com"
```

A kapcsoló az `ldapadd` paranccsal is használható.
A fenti példában szereplő ***ldaps://*** URL megadása kötelező.
A `-x` kapcsolóval egyszerű, nyílt szöveggel történő hitelesítést választottam, így szükségszerűen TLS titkosítással védett kapcsolatot létesítek a kiszolgálóval. A múltkor már megmutattam, hogyan lehet TLS-kapcsolatok fogadására beállítani az LDAP-kiszolgálót.
Nehéz lenne elhallgatnom azt a tényt, hogy LDAP használatakor a felhasználók jelszavainak a kezelése okozza a legtöbb gondot. Ha viszont a felhasználók hozzáférést kapnak az `ldappasswd` parancshoz, akkor helyi ***/etc/ldap.conf*** fájlokkal és parancsfájlokkal vagy megfelelő felületet biztosító programokkal egyszerűen megváltoztathatják a saját jelszavukat.
A más operációs rendszert futtató felhasználók jelszavait azonban központilag kell kezelni, és – hacsak nem telepítünk LDAP ügyfélprogramot az ő gépükre is – a felhasználók minden jelszóváltoztatás alkalmával kénytelenek felvenni a kapcsolatot a rendszergazdával. A Microsoft Windows alapú munkaállomásokhoz a Samba úgy is beállítható, hogy a felhasználók számára lehetővé tegye LDAP-jelszavuknak a Windows beépített jelszókezelőjével történő megváltoztatását (lásd az „OpenLDAP mindenütt” című írást a Linuxvilág 2003. januári számában).

Hozzáférés-vezérlés

Gyakorlatilag minden olyan tennivalót átvettünk, vagy legalábbis érintőlegesen megemlítettünk, amelyet egy OpenLDAP alapú LDAP-kiszolgáló üzembe helyezéséhez el kell végezni (kivéve a különféle kiszolgálóalkalmazások LDAP alapú hitelesítésre való átállításának cseppet sem lebecsülendő feladatát). Ha erőteljes biztonságra törekszünk, márpedig ha az egészbe belekezdtünk, csakis ez lehet a cél, akkor még egy dologról szót kell ejtenünk: az OpenLDAP hozzáférés-vezérlési listáiról (ACL).
Ahogy a `slapd` démonra vonatkozó beállítások többségének, úgy az ACL-eknek a megadása is a ***/etc/openldap/slapd.conf*** fájlban történik. Az LDAP kapcsán talán nem meglepő, hogy az ACL-ek kezelése (is) okozhat némi fejtörést, és a kívánt eredmény eléréséhez bizony szükség lesz némi kísérletezésre. A *2. lista* az ACL-ek megadására mutat példát.
Az ACL-ek részletes ismertetése a slapd.conf(5) súgóoldalon található meg, de működésük alapjait a 2. kódrészlet alapján is fel lehet ismerni. Minden védeni kívánt LDAP-elemhez meg kell adnunk, hogy pontosan kik kapnak hozzáférést, illetve ez milyen szintű legyen. Egyébként a teljes ACL-t egyetlen sorban meg lehetne adni, de a hagyomány az, hogy minden "***by...***" állítást külön sorba írunk. A `slapd` elég okos ahhoz, hogy tudja, az "***access to***" karakterlánc a következő ACL kezdetét jelzi. Helyszűke miatt részletesen nem ismertethetem az ACL-ek

2. lista ACL-ek a /etc/slapd.conf fájlban

```
access to attrs=userPassword
  by dn="cn=ldapproba,dc=proba,dc=org" write
  by self write
  by * compare

access to *
  by dn="cn=ldapproba,dc=proba,dc=org" write
  by users read
  by * auth
```

írásmódját, így csak néhány dolgot emelnék ki. Először is, az ACL-ek feldolgozása fentről lefelé történik, és az első találat jut érvényre, mintha szűrők kupacáról lenne szó. Fontos tehát, hogy a különleges ACL-eket az általánosabbak fölé helyezzük. A 2. listában például egy olyan ACL-t láthatunk, amely a `userPassword` jellemző elérését korlátozza, ezt pedig egy a teljes LDAP-adatbázisra vonatkozó szabály követi.
A `userPassword` ACL-t előre helyezve engedélyezhetjük a felhasználóknak saját jelszavuk módosítását (`access to attrs=userPassword by self write`), de ez a lehetőség kivételt képez ahhoz képest, hogy a felhasználók bármit olvashatnak (`access to * by users read`).
Ugyancsak hangsúlyoznom kell, hogy a hozzáférési szintek hierarchiába szerveződnek. A lehetséges szintek a `none`, az `auth`, a `compare`, a `search`, a `read` és a `write`, ahol a `none` a legalacsonyabb szint, a `write` pedig a legmagasabb, és mindegyik szint magába foglalja a nála alacsonyabbak által megadott jogokat is. Az első egyezés érvényre jutása és a hozzáférési szintek egymást való tartalmazása az a két jellegzetesség, amelyet az ACL-ek kezelése kapcsán meg kell érteni, már csak azért is, mert gondosan ügyelni kell arra, hogy az ACL-ek ne adjanak a kívántnál bővebb jogokat, de a munkához szükséges hozzáféréseket biztosítsák.

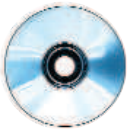
Összegzés

Az LDAP az általam az utóbbi időkben látott egyik legbonyolultabb megoldás. Ha saját elvárásaink szerint szeretnénk munkára bírni, akkor fel kell készülnünk arra, hogy rengeteg próbálgatásra lesz szükségünk, naplókban kell hosszasan turkálnunk, és mind az LDAP-kiszolgáló, mind a segítségével hitelesítést végző alkalmazások oldalán idegölő finomhangolást kell végeznünk. Ha azonban figyelembe vesszük, hogy milyen rugalmas, nagy teljesítményű és széles körben támogatott hitelesítő és címtárkezelő eszközt kapunk, máris enyhülnek az álmatlan éjszakák kínjai. Remélem, hogy írásaim segítségével sikerül ezt a munkát véghez vinni, vagy legalábbis közelebb jutni a kitűzött célhoz.

A cikkhez tarozó Kapcsolódó címek megtalálhatóak az 53. CD Magazin/LDAP könyvtárában.

Linux Journal 2003. szeptember, 113. szám

Mick Bauer (mick@visi.com)
Biztonsági szakember, a Linux Journal biztonsági témákkal foglalkozó szerkesztője, biztonsági tanácsadó a Minnesota állambeli Minneapolisban található Upstream Solutions LLC Inc.-nél. Mick szabadidejében a gyermekeivel fogócskázik vagy zenél.



NSA fokozott biztonságú Linux

Rendszered kötelező, finomhangolt hozzáférés-beállításaival azt is korlátozni tudod, hogy a rendszergazda mit tehet.

Az NSA fokozott biztonságú Linux gyökerei a korábban megjelent, nagy bizalomnak örvendő operációs rendszerig (DTOS) és a Flaskig (Flux advanced security kernel – Flux fokozott biztonsági rendszermag) nyúlnak vissza. A DTOS projekt az amerikai National Security Agency (NSA) és a Secure Computing Corporation (SCC) 1990-es évek elejiközepi együttműködésének az eredménye. A cél egy, a standard biztonsági módszereknél erősebb biztonsági eljárásokkal bíró operációs rendszer készítése volt. A Flask-szerkezet az NSA, az SCC és a University of Utah Flux projektjének a végeredménye, amelyet „kibővítettek, hogy jobban megfeleljen a változó biztonsági előírásoknak" (***Stephen Smalley*** „Flask: Flux Advanced Security Kernel", NAI Labs, 2000. december 26.; ➲ http://www.cs.utah.edu/flux/flask).

Az SE Linux megköveteli a hozzáférés-beállítást (Mandatory Access Control – MAC), míg egy átlagos Unix-rendszer tetszés szerinti hozzáférés-beállítással dolgozik (Discretionary Access Control – DAC). A DAC esetében a felhasználó szabja meg, hogy az egyes elemeknek mekkora hozzáférésük legyen az egyes területeken. Egy Unix-rendszeren például alkalmazhatjuk a `chmod` parancsot a saját könyvtárrendszerünk engedélyeinek megváltoztatáshoz. A MAC esetében a hozzáférés-beállítást egy magasabb szinten lévő felhasználó szabja meg, ő határozza meg a biztonsági beállításokat, amelyek megadják az egyes elemek jogait. Ha létezik egy olyan szabály, amelyik megakadályozza, hogy Bob megnézze Alice otthoni fájljait, és Alice kiadja a `chmod 777` utasítást az otthoni könyvtárában, Bob ennek ellenére sem tudja megnézni őket.

A MAC-et használva a programok minimális jogosultságokkal futnak, egy „megtámadott" folyamat nem tud hozzáférést adni a saját forrásaihoz olyan más folyamatok számára, amelyeknek ez eredetileg nincs engedélyezve. Ez mérsékli a károsodás mértékét abban az esetben, ha egy démon megsérülne. A biztonsági beállítások számos tényezőn múlnak, például a felhasználó jogosultságain, a futtatott program jellegén, azon, hogy mennyire megbízható a program, és milyen a kezelt adatok titkossági szintje vagy épsége.

Mi az SE Linux?

Az SE Linux egy rugalmas, finomhangolt hozzáférés-beállítás a Linux-rendszermagban, amely jelenleg az LSM-vázat alkalmazza (lásd Linux Journal, 2002. november, ***Greg Kroah-Hartman*** „Using the Kernel Security Module Interface"). A jelenlegi felhasználáskor az LSM-csatoló csak a korlátozó hozzáférés-beállítást támogatja, ezért ha a Unix-engedélyek leállítanak egy műveletet, azt az SE Linux sem engedélyezheti. Az SE Linuxot általában további korlátozások életbe léptetésére használják olyan rendszereken, ahol Unix-engedélyek élnek, és ez magától is meglehetősen jól tudja a szükséges hozzáférés-beállításokat. Az SE Linux egy rendszermagfoltból és más segédprogramokhoz tartozó foltok áll, ilyen például a `login` és `cron`. Az NSA felel a hivatalos változatokért. Néhány, nem az NSA-ban dolgozó személy is hozzájárult kódokkal a projekthez. A csomagokat folyamatosan fejlesztik a Debian üzembiztos és fejlesztői változataihoz. Mivel az SE Linux a GPL felhasználási szerződés hatálya alá esik, bárki hozzáírhat és saját módosításokat készíthet. Az SE Linuxot 2.4.19-es és frissebb rendszermagokon lehet használni, és e cikk írásakor – 2003 májusában – a 2.5-ös rendszermagra is fejlesztik.

Miért van szükség a módosított alkalmazásokra?

Mint azt már korábban említettük, az SE Linux egy rendszermagfoltból és módosított segédprogramokból áll. A módosított segédprogramok teszik lehetővé, hogy a rendszer minden fájlja helyes biztonsági tartalommal bírjon. A segédprogramoknak módosított változatai is léteznek, például a `login`, a `cron` és a `logrotate`, valamint olyan programok, mint a `ps` és az `ls`. A `login` esetében például létfontosságú a helyes biztonsági környezet, amikor felhasználó bejelentkezik a rendszerbe. Ha ez nem él, akkor elképzelhető, hogy a felhasználó egyáltalán nem tud belépni. A `login` csomag telepítését a „Hogyan kezdjük?" fejezet írja le az SE Linux HOWTO-jában (lásd a ***Kapcsolódó címeket***), és ez nem is tartozik e cikk keretei közé. Ha azonban az SE Linux telepítésekor elfelejtjük felrakni a `login` csomagot, ez azt eredményezi, hogy nem lesz megfelelő kód kiadva a terminálnak (terminal device), ahol az újraindítást követően bejelentkezne, és ez lehetetlenné teszi a bejelentkezést. A módosítatlan `login` program szintén futtat egy héjprogramot a biztonsági környezetben, amely letiltja a fájlok hozzáférését a felhasználó saját könyvtárában. Például a `login` és `cron` foltjai megmondják a rendszermagnak, hogy melyik biztonsági környezetet alkalmazza. Ezek tényleges végrehajtását a rendszermag végzi. A címkézés létfontosságú, ezért van szükség néhány módosított programra. Mindenki elkészítheti a saját biztonsági irányelveit, amelyek a módosított csomagok telepítése nélkül is alapszintű védettséget biztosítanak, de az alapbeállítás kifinomultabb biztonsági rendszert eredményez.

Gyakran használt kifejezések

Az alábbi kifejezésekkel mindig találkozhatunk, ha az SE Linux leírását vagy a levelezőlisták hozzászólásait olvassuk. Fontos, hogy ezekkel minél jobban megismerkedjünk, még mielőtt telepíteni próbálnánk az SE Linuxot, ezáltal ugyanis a későbbi munkánkat is megkönnyítjük.

- Tartomány
  Egy tartomány részletezi, hogy az egyes folyamatok mit tehetnek meg, és mit nem, illetve azt, hogy egy folyamat milyen műveleteket hajthat végre eltérő típusokon. Ha a `felhasznalo_t` tartományban vagyunk (megkötések nélküli felhasználó tartomány) és a `ps aux` parancsot futtatjuk, akkor csak a `felhasznalo_t` tartományban futó programokat látjuk. Egyéb példák a tartományra: a `sysadm_t`, a rendszergazdai tartomány és az `init_t`, amely az a tartomány, amelyben az `init` fut. A `passwd_t` tartományban a `passwd` programot egy jogosultságokkal nem rendelkező felhasználó futtathatja.

- Jogosultságok
  A jogosultság határozza meg, hogy mely tartományokat lehet használni. Az irányelveket tartalmazó adatbázis (policy database) határozza meg az egyes felhasználók által hozzáférhető tartományokat. Ha egy adott felhasználónak nincs engedélye (a beállításfájlban) belépni egy tartományba, akkor letiltják. Két példa a jogosultságokra: az átlagos, jogosultságokkal nem rendelkező felhasználói szerep (`felhasznalo_r`) és a rendszergazdai jogosultság (`sysadm_r`).
  Nézzük a következő példát: ahhoz, hogy egy felhasználó a `felhasznalo_t` tartomány alól egy `passwd` parancsot hajtson végre, a `felhasznalo_r` jogosultságú felhasználónak kell begépelni: `passwd_t`; – ez a vonatkozó beállításfájlban van megadva. Ezenkívül más tartományátmeneti szabályokat is fel kell állítani, ezek bővebb ismertetéséről azonban terjedelmi okokból le kell mondanunk. Ez a hozzáadott kód szabja meg, hogy egy felhasználó „felhasználói” (`user_r`) jogosultságokkal elérhesse a `passwd_t` tartományt ahhoz, hogy lefuttassa a `passwd` parancsot. Más kérdés az, hogy vajon a régi tartománynak van-e jogosultsága átlépni az új tartományba.
  Most, hogy meghatároztuk a tartományokat és a szabályokat, összehasonlíthatjuk az SE Linuxot és a Unix `uid`-ot (felhasználói ID). Ha a rendszergazda ural egy programot, amelynek az Unix engedélye 4777 (ez a programot „setuid” rendszergazdává teszi), akkor a rendszer bármelyik felhasználója végre tudja hajtani azt a programot, biztonsági hibát okozva ezzel. Az SE Linuxon azonban ha egy művelet átlép egy védett tartományt, és ha a művelet jogosultsága nem engedélyezett azon a tartományon, akkor a program nem fog futni. Az SE Linux-rendszereken minden művelet egy olyan tartományon fut, amelyik megszabja, hogy az egyes műveletek milyen hozzáférési joggal rendelkeznek.
- Azonosító
  Az SE Linux alatt az azonosító nem ugyanazt jelenti, mint a hagyományos Unix `uid`, amit az olvasók talán ismernek is. Az SE Linux alatt az azonosítók annak a biztonsági környezetnek részei, amelyek azt irányítják, hogy mit tehet meg az ember, és mit nem. Elképzelhető, hogy egy SE Linux azonosítónak és egy hagyományos Unix bejelentkezési névnek ugyanaz a szöveges megjelenítése (és legtöbbször a könnyebbség kedvéért ez igaz is), de fontos megértenünk, hogy két különböző dologról van szó. Alapértelmezés szerint ugyanazok, ha a kérdéses SE Linux azonosító létezik. Ebből kifolyólag, ha `faye` felhasználóként jelentkezem be az SE Linux-rendszerbe, és az irányelv-adatbázisban szerepel `faye`, akkor az én műveleteimhez a `faye` azonosítót rendelik.
  Vegyük például a `su` parancsot a Unix-felhasználó ID-k és az SE Linux azonosítók különbségeit bemutatandó.
  A `su` futtatása nem változtatja meg a felhasználói azonosítót az SE Linux alatt, de megváltoztatja az `uid`-ot, ahogy azt a nem SE Linux-rendszeren is tenné. Ha a `faye` felhasználó az SE Linux-rendszeren gépelné be, hogy `su` ahhoz, hogy rendszergazda lehessen, majd lefuttatná az `id` parancsot, amely kiadja az ő biztonsági környezetét és egyéb adatait, akkor láthatná, hogy az azonosítója továbbra is `faye`, nem pedig root, az `uid` azonban megváltozott.
  Hogy ezt jobban érzékeltessem: ha egy átlagos felhasználó a `faye` loginnal futtatja az `id` parancsot, akkor a következő biztonsági környezetet látja:

```
uid=1000(faye) gid=1000(faye)
groups=1000(faye)
context=faye:felhasználó_r:felhasználó_t
sid=45
```

  Ebben az esetben a biztonsági környezet azonosító része `faye`. Láthatjuk, hogy az `uid` 1000. Tegyük fel, hogy `faye` ezután beüt egy `su` parancsot, és ismét futtatja az `id` parancsot. Most ezt látja:

```
uid=0(root) gid=0(root) groups=0(root)
context=faye:felhasználó_r:felhasználó_t
sid=453
```

  A várttól eltérően az azonosító nem változott rendszergazdára, de az `uid` 0-ra változott. Ha azonban a `faye` felhasználónak engedélye van a rendszergazda jogosultság alkalmazására vagy `sysadm_r`-re, akkor ezt megteheti: a konzolnál történő bejelentkezéskor megadja, hogy ő szuperfelhasználó kíván lenni, vagy a `newrole -r` parancsot adja ki – erről még szólunk a továbbiakban. Ha ezt követően újból futtatja az `id` parancsot, ezt látja majd:

```
context=3Dfaye:sysadm_r:sysadm_t
```

  Így az azonosító ismét ugyanaz marad, de a jogosultság és a tartomány (a második és harmadik mező) megváltozik. Az azonosító ilyen módon történő megtartása ott hasznos, ahol a felhasználó megbízhatósága megkövetelhető. Ez létfontosságú a rendszer biztonsága szempontjából is, mivel a felhasználó azonosítója határozza meg, hogy mely szerepeket és tartományokat lehet használni. Egy átlagos Unixon, ha az embernek van egy `setuid` vagy egy `setgid` programja, amely nem mindenki álltal futtatható, ennek elindíthatóságát nemcsak a bejelentkezett felhasználó engedélyei szabják meg, hanem az is, hogy mely felhasználó felé végeztünk legutoljára egy `su` keresést. Ez a megkötés nem létezik az SE Linux alatt, mivel az az egyén azonosítóját minden művelet végrehajtásakor rögzíti. Ha egy tartományon nincs engedélyünk setuid/setgid program futtatására, akkor nem fogjuk tudni futtatni, mégha egy `su`-val a rendszergazdává váltunk. A jogosultságod határozza meg, hogy mely tartományba van engedélyed belépni, az általad megadható jogosultságokat pedig az azonosítód szabja meg. Ebből kifolyólag az azonosító közvetve irányítja az általad használható tartományok listáját.
- Típus
  Minden tárgy egy típus alá van sorolva, ez a típus határozza meg, hogy mi férhet hozzá az adott tárgyhoz.
  A tárgyak jelen esetben fájlok, könyvtárak, foglalatok (sockets) és más műveletek. Egy típus elgondolásában a tartományhoz hasonlít, a különbség az, hogy a tartomány csak a műveletekre vonatkozik. Másként fogalmazva: a tartomány egy olyan típus, amelyik a műveletekre vonatkozik.
- Átmenet
  Egy lekért művelet során megváltozó biztonsági környezet átmenetére vonatkozik. Az átmenet két csoport egyikébe tartozik. Az első csoport a tartományműveletek átmenete. Amikor egy adott programot futtatunk, az átmenet indulhat a jelenlegi tartományból egy új tartományművelet felé. Ennek bemutatására a `newrole` parancsot használom.

A `newrole` parancs a szereped megváltoztatását szolgálja, mondjuk a `felhasznalo_r`-ről a `sysadm_r`-re – feltéve, hogy `sysadm_r`-ként van hozzáférésed. Ha `felhasznalo_r`-ként indulsz (ez az átlagos, jogosultságokkal nem rendelkező felhasználó), és lefuttatsz egy `newrole -r sysadm_r`-t, hogy megváltoztasd a `sysadm_r`-t, a rendszergazdai szerepet, átmenetet képezel a `felhasznalo_t` tartományból a `newrole_t` felé (ez az a tartomány, ahol az új jogosultság művelet futni fog), és onnan pedig a `sysadm_t` tartomány felé.
A második átmeneti kategória a fájltípusok átmeneti jellege, amikor fájlt nyitunk egy adott könyvtár alatt. Ha egy felhasználó egy fájlt hoz létre a saját könyvtárában, akkor annak a fájlnak a címkéje `user_home_t` lesz. De ha ugyanez a felhasználó egy fájlt a */tmp*-ben hoz létre, akkor az a fájl a `user_tmp_t` címkét kapja. A `user_tmp_t` a */tmp* típusból lett elvonva, amely a `tmp_t`, és abból a tartományból, ahol a fájlt létrehozták, ez a `user_t`. Amikor a felhasználó létrehoz egy fájlt a */tmp* alatt, átmenet zajlik le a `user_tmp_t` típus felé.

- Irányelvek
  Egy irányelv határozza meg, hogy mely típusok mely műveleteket hajthatják végre az egyes tartományokban. Minden démonnak megvan a saját irányelve és a névadási szabály az ***démon-name.te***, ***postfix.te***, ***apache.te*** stb. Egy SE Linuxot futtató gép rendszergazdájaként tetszés szerint szerkesztheted az irányelv fájlokat. Az irányelv-adatbázis az irányelvek forrásfájljainak az összesített listája, amit a rendszermag tölt be indításkor.

SE Linux rendszeren a `spasswd` programot használjuk a jelszó megváltoztatására. A `spasswd` gyakorlatilag a burkolója (wrapper) a Linux-rendszeren használt `passwd`-nek, ez biztosítja, hogy a `passwd` program a helyes tartományon fusson. Ugyancsak ez biztosítja azt is, hogy az SE Linux-azonosítód megegyezzen az egyébkénti Unix-azonosítóddal. Korábban említettem, hogy az átlagos Unix-felhasználó ID-k sokban különböznek az SE Linux-azonosítóktól, de akkor most, amikor a `spasswd`-t futtatom, miért kell egyezniük? A `spasswd` követeli meg, hogy az SE Linux-azonosítóneved megegyezzen a Unix-azonosítóddal. Emlékezz csak, az SE Linux-rendszeren az azonosítód az egyetlen módja annak azonosítására, hogy ki is vagy. Tehát ha legutóbb nem te voltál a megadott Unix-felhasználó, akkor nem tudod megváltoztatni a jelszót.
Ha rendszergazda vagy (`sysadm_r`), egy másik felhasználó jelszavának a megváltoztatásra a `sadminpasswd` programot használhatod. A `sadminpasswd` esetében nem él ez az azonos felhasználónév/azonosító megkötés, mint a `spasswd`-nél, de a `sadminpasswd` csak `sysadm_t`-ként futtatható.

Megengedő és megkötő módok

Az SE Linux kétféle üzemmódban futtatható: megengedőben vagy megkötőben. A megengedő módot hibakeresés céljából használjuk, mivel ott minden naplózódik, de az SE Linux valójában nem köti magát az irányelveidhez. A felhasználó továbbra is tud rendszergazdaként végrehajtani dolgokat, csakúgy, mint egy átlagos Linux-rendszeren. Amíg meggyőződsz róla, hogy minden irányelv kielégítően működik, legjobb a gépet megengedő módban futtatni. Címkéket adunk a rendszerben található tárgyaknak, de semmit sem kényszerítünk ki. A megkötő mód a már megadott irányelveket használja, például a hozzáférés-beállításokat. Csak akkor indítsd a gépedet megkötő módban, ha meggyőződtél róla, hogy az rendesen működik, és már egy ideje rendesen futott megengedő módban. Ne feledd, hogy ha a rendszermagba nincs befordítva a fejlesztői támogatás, akkor itt nem futtahatod megengedő módban a rendszert. Ha a rendszermagban be van kapcsolva fejlesztői támogatás, akkor a gép megengedő üzemmódban indul, és kézzel kell megkötő módra kapcsolnod. Ezt könnyen megteheted egy indító parancsfájl segítségével. A másik megoldás, hogy kiépítesz egy ilyen utat:

```
/etc/rc.boot/avc and /sbin/
avc_toggle
```

Újabb megoldás: írd be a rendszermag parancssorába, hogy `enforcing=1`. Az `avc_toggle` parancsot használhatod a megengedő és a megkötő mód közti változtatásra, és `avc_enforcing` parancsot használhatod annak megállapítására, hogy megkötő módban vagy-e éppen.

Hogyan tovább?

E cikk olvasása után remélhetőleg kedvet kaptál az SE Linux kipróbálásához. Szándékosan hagytam ki a telepítési útmutatót, mivel RPM-ek, az úgynevezett source tarballs vagy Debian-csomagok segítségével is telepíthetsz. Az alapvető lépések leírása egy külön cikket is megtöltene. Egy sereg új dolgot meg lehet tanulni a telepítés előtt, közben és után, és az új felhasználók gyakran összezavarodnak. De ha – még mielőtt bármibe is belefognál – elolvasod a ***Kapcsolódó címek*** részben megnevezett dokumentumokat és megismerkedsz a gyakran használt fogalmakkal, az egészet valamivel könnyebbnek találod majd. Ha elakadsz, izzítsd be kedvenc IRC-ügyfeledet, és menj a #selinux csatornára a ➲ http://irc.debian.org alatt, vagy iratkozz fel az SE Linux-levelezőlistára.

Linux Journal 2003. augusztus, 112. szám

Faye Coker
Jelenleg szabadúszó rendszergazda. Gyakran futtat ISPs-rendszereket és állít át kiszolgálókat Linuxra. Dolgozott Európában és Ausztráliában. A kelleténél többször kérdezik meg tőle Linux-konferenciákon, hogy „eltévedtél?”.

KAPCSOLÓDÓ CÍMEK

Flask (Flux Advanced Security Kernel)
➲ www.cs.utah.edu/flux/flask
Getting Started with SE Linux HOWTO
➲ sourceforge.net/docman/display_doc.php?docid=15285&group_id=21266
NSA Official SE Linux oldal ➲ www.nsa.gov/selinux
NSA SE Linux GYK ➲ www.nsa.gov/selinux/faq.html
NSA SE Linux White Papers
➲ www.nsa.gov/selinux/docs.html
SE Linux levelezőlista ➲ www.nsa.gov/selinux/list.html
SE Linux archívum ➲ marc.theaimsgroup.com/?l=selinux
SourceForge SE Linux Project oldal
➲ http:/sourceforge.net/projects/selinux

ChessBrain

Ha egyetlen számítógép is megver sakkban, mi lesz, ha 646 fog össze ellened?

Idén májusban 646 gép működött együtt egyetlen sakkparti lejátszásában. Ez volt az első alkalom, amikor ilyen eredményt könyvelhettek el, és ezt a Linux, a nyílt programkód és 37 ország több száz programozójának együttes ereje tette lehetővé.

A ChessBrain (➲ http://chessbrain.net) egy osztott számítási program, ami a hálózaton elérhető gépek kihasználatlan számítási kapacitását aknázza ki nagy számításigényű feladatok megoldására. A ChessBrain-rendszer sakkjátszmák lejátszására összpontosít, de a háttérrendszer más játékokhoz vagy nem játék jellegű feladatokhoz is átalakítható.

Képzeljünk el egy teljesen szokványos játszmát, azzal a különbséggel, hogy az ellenfél minden lépése után telefont ragadunk és elkezdjük fölhívni a barátainkat, hogy adjanak tanácsot. Elmagyarázzuk az állást az elsőként hívott barátunknak, és megkérjük, hogy hívjon vissza, ha van valami ötlete. Azután másvalakitől kérdezzük meg, hogy aggódnunk kell-e egy küszöbön álló támadás miatt, és őt is megkérjük, hogy hívjon vissza, ha megvan a válasz. Valahogy így sakkozik a ChessBrain.

A ChessBrain két része a SuperNode nevű, Linux alapú kiszolgálóalkalmazás és a PeerNode nevű ügyfélprogram. A SuperNode egy hálózati játékkiszolgálóhoz csatlakozik, ami lehetővé teszi, hogy a bejelentkezett tagok egymás ellen játsszanak, kihívják a ChessBraint, vagy figyelemmel kísérjék a ChessBrain és pillanatnyi ellenfele játszmáját. A ChessBrain játék közben megvizsgálja az állásokat, lehetséges lépések százait küldi el feldolgozásra a távoli PeerNode-ügyfelekhez, begyűjti a válaszokat, feldolgozza őket, és meglépi a legjobb lépést. A ChessBrain hálózatba kötött gépek folytonosan változó halmazaként létezik. Ez mind filozófiai, mind tudományos szempontból nézve gyönyörű.

A ChessBrainnel 2001 nyarán kezdtem el foglalkozni mint osztott számítási kísérlettel. Az év végére elkészült egy működő prototípus, szükségem volt tehát egy helyre a kiszolgálóprogram számára. Régi barátom, ***Walter Howard***, a HackerWacker honlap (➲ http://hackerwacker.com) gazdája felajánlotta a saját T1 vonalát a kiszolgáló elhelyezésére.

2002. június 9-én a ChessBrain föltűnt a Slashdoton, a kedvező hírverés hatása pedig több száz új PeerNode-ot futtató fehasználó megjelenésében mutatkozott. Egyikük, ***Gavin Roy,*** a bteg hálózat (➲ http://www.bteg.net) tulajdonosa, felajánlotta, hogy ingyen helyet biztosít a SuperNode-kiszolgálónak. Június 27-én Gavinnel ebédeltem, és átnyújtottam ennek a jószerivel idegen fickónak egy SuperNode-kiszolgálót egy Pentium III gépen. A ChessBrain gazdagabb lett egy kiszolgálóval, én pedig egy baráttal, Gavin ugyanis a ChessBrain egyik fontos támogatója lett. Áthelyeztem a SuperNode-kiszolgálót Gavin oldalára, Walt pedig tovább üzemeltette az eredetit tartalék- és kísérleti kiszolgálóként.

Az ezt követő hónapok során hihetetlen mértékű figyelem irányult ránk. Nem úgy tűnt, mintha bárkit is zavarna, hogy a ChessBrain tulajdonképpen nem tud sakkozni. 2002 első nyolc hónapja a SuperNode-kiszolgálón végzett munkával és a PeerNode-ügyfél Microsoft Windows és Apple Mac OS X rendszerre történő átültetésével telt.

Amikor a kiszolgáló és az ügyfelek jól működtek, a hangsúly arra tevődött át, hogy a ChessBrain képes legyen sakkozni. A hollandiai wbec-ridderkek (➲ http://www.wbec-ridderkek.nl) honlap közel 200 szabadon elérhető sakkprogramot sorol föl. Átnéztem néhányat, viszonylag tiszta és több operációs rendszeren fordítható kódot keresve. Tökéletes programnak bizonyult a Beowulf, amit ***Colin Frayn*** írt, aki akkor az angliai Cambridge Egyetem doktorandusza volt. Váltottunk jó néhány levelet, és Colin beszállt a fejlesztésbe. Végig hálózaton keresztül folytattuk az együttműködést, elektronikus levelek és csevegőüzenetek (instant messaging) segítségével, és elkezdtük a szükséges változtatásokat. Colin osztott számításhoz igazította a sakkprogramját, én pedig módosítottam a SuperNode és PeerNode programokat, hogy használhassák a sakkprogramot. A London és Los Angeles közötti időeltolódás éppen megfelelt nekünk. Üzentem Colinnak délután háromkor, és utána a nap folyamán. Az én délutánom közepe felé Colin lefekvéshez készülődött, és én végigdolgoztam az ő éjszakáját. Összeomlás előtt küldtem neki valami visszajelzést. Ez az egész napos ciklus hónapokig folytatódott.

1. kép Az első ChessBrain-kiszolgálók

Colin eredeti Beowulf sakkprogramjából két sakkozó összetevőt hozott létre BeoServer és BeoClient néven. Úgy alakította ki ezt a párost, hogy a ChessBrain keretein belül játszhassanak le sakkjátszmákat. 2002. december 22-én a ChessBrain lejátszotta első osztott számítással végzett sakkpartiját. 2003 januárjára a ChessBrain-közösség 62 gépet üzemeltetett és rendszeresen tesztelte az új programváltozatokat.

Áttekintés

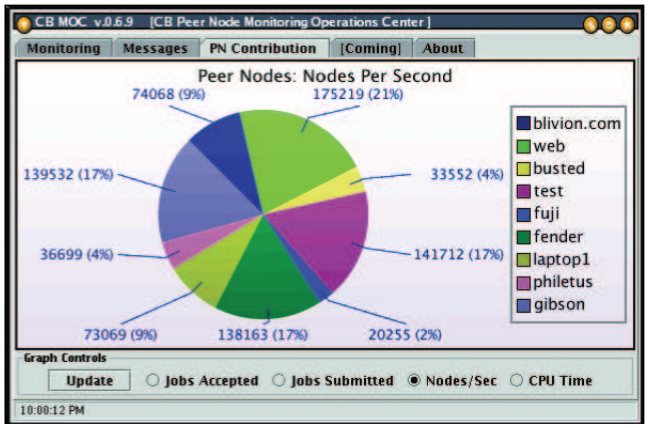
A SuperNode és a PeerNode többszálas C++-program, amit GCC-vel fordítottunk Red Hat Linux 7.1, 7.2 és 8.0 rendszeren. Az elsődleges SuperNode-kiszolgáló Slackware 8.0 alatt fut a bteg hálózat észak-kaliforniai telepén (*1. kép*).
Mivel az alkalmazások erősen többszálasítottak, sok időt kellett a szálakkal kapcsolatos gondok megoldására fordítanom. A GDD és a DDD mellett egyedi naplózással kezeltem a hibakeresési feladatokat. A fejlesztési folyamat korai szakaszában a Perl-parancsfájlok különösen alkalmasnak bizonyultak az új szolgáltatások kipróbálására és a program terheléses tesztelésére. 12 gépem van otthon, ezek egy seregnyi, a helyi kiszolgálót bombázó Perl-parancsfájllal tekintélyt parancsoló teszteszközt alkottak.

XML, SOAP és webszolgáltatások

A fejlesztés kezdetén világos volt, hogy a SuperNode kiszolgálónak kapcsolatot kell tartania más kiszolgálókkal. Akkoriban az XML megfelelő megközelítési módot jelentett, és később az XMLRPC (http://www.xmlrpc.org) további előnyöket nyújtott. Az egyszerű objektumelérési protokoll (Simple Object Access Protocol, SOAP) a más kiszolgálókkal beszélő kiszolgálók igényének kielégítése irányába fejlődött. A kedvezőbb együttműködés lehetősége bátorított arra, hogy a SOAP legyen a SuperNode-kiszolgáló és a PeerNode-ügyfelek közötti kapcsolattartás eszköze. Kívülről nézve a SuperNode úgy működik, mint egy webkiszolgáló SOAP alapú csatlakozófelületekkel. Bár a SuperNode-kiszolgáló GET- és POST HTTP-üzeneteket is használ, a POST a gyakoribb. A SuperNode HTTP és XML alapú SOAP-kéréseket fogad, feldolgozza őket, és HTTP-csomagokat küld vissza, beágyazott SOAP kihelyezésekkel.
A SuperNode és a PeerNode feldolgozza a SOAP-kéréseket, és a parancsokat egy belső parancselosztónak továbbítja, ami biztosítja, hogy a megfelelő parancskezelő dolgozza fel a kérést. A SuperNode-ban a leggyakoribb kérések a PeerNode-ügyfelektől érkeznek; a PeerNode-nak kapcsolódnia kell, hogy lekérhesse a következő feldolgozási egységet. A feldolgozási egység olyan XML-blokk, ami egy játékállást ír le, és utasításokat ad arra nézve, hogy az állást hogyan kell elemezni.
A PeerNode egy teljes sakkprogrammotort tartalmaz, amit statikus könyvtárként fordítunk. Amikor a PeerNode megkapja a feldolgozási egységet, értelmezi a SOAP-választ, kibányássza a feladatra vonatkozó adatokat és utasításokat küld a sakkmodulnak.
Ezután a SuperNode-kiszolgáló a pillanatnyi állást elküldi a külső BeoServer folyamatnak. A folyamatközi kapcsolattartás két cső segítségével valósul meg. A közeljövőben a BeoServer terveink szerint külön gépre kerül, és UDP 1000Base-T ethernet alkalmazására állunk át.

Biztonság

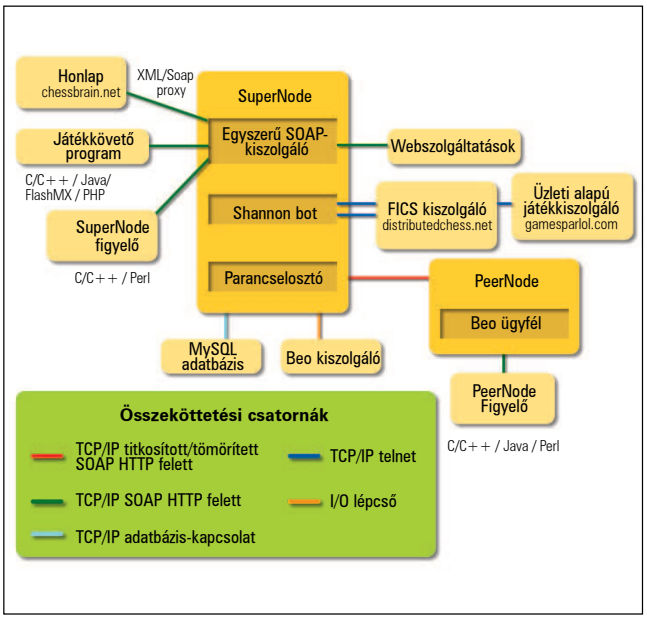
A biztonságos és hiteles üzenetváltás lényeges a ChessBrain számára. Egy rosszindulatú felhasználó egy meghamisított eredménnyel elronthatná a játékot és kínos helyzetet teremthetne. Az érzékeny adatokat a továbbítás során az AES Rijndael nevű fejlett titkosítási szabvány (Advanced Encryption Standard) védi. Az AES a belga ***Joan Daemen*** és ***Vincent Rijmen*** által kifejlesztett változó blokkhosszúságú, szimmetrikus titkosítási algoritmuson alapul, aminek a kiöregedő DES és háromszoros DES szabvány kiváltása a célja.
Mielőtt a Rijndaelt felfedeztük volna, a Blowfish szimmetrikus kódolót használtuk, de a PeerNode Mac OS X rendszerre történő átültetésekor az általunk használt Blowfish-megvalósításban gondok merültek fel a bájtsorrend körül. Az AES algoritmus működése nem függ a bájtsorrendtől, így az adott helyzetben ez volt a legjobb megoldás.
A PeerNode eredeti felépítése szerint két külön folyamatban futtatta az ügyfélprogramot és sakkprogramot. A PeerNode indította a sakkprogramot, és a szabványos kimenet átirányításával hozott létre laza kapcsolatot. Eredetileg el akartuk kerülni a sakk-kód beültetését a PeerNode-ügyfélbe, hogy a program későbbi változataiban könnyen és gyorsan lecserélhessük. Később biztonsági megfontolásból statikus csatolásra tértünk át. A gondot az okozta, hogy lehetett olyan sakkproxy programot írni, ami a PeerNode és a valódi sakkprogram között helyezkedik el. Ez könnyen lehetővé tette volna az eredmények megváltoztatását, mielőtt a SuperNode-kiszolgálóra



2. kép Java alapú CBMoc

3. kép Valós idejű OpenGL-leképezés

kerülnek. A statikus csatolás mellett két fontos előnye miatt döntöttünk: az egyik a nagyobb biztonság, a másik pedig a függvényalapú adatátadás az I/O alapú helyett.
A Slashdotról kiinduló érdeklődéshullám szükségessé tette a ChessBrain sávszélességigényének a korlátozását. Ebből a szempontból nézve, bár a SOAP számos előnyös tulajdonsággal bír, a mérete némi kívánnivalót hagy maga után.
Ma már a ***zlib*** tömörítőkönyvtárat (➲ http://www.zlib.org) használjuk a titkosítás előtt a SOAP alapú üzenettovábbítás méretének csökkentésére. A tömörítés és a titkosítás korlátozza az együttműködési lehetőségeket, ugyanakkor az XML-tömörítési szabvány (➲ http://www.w3.org/TR/xmlenc-core) egy másik megközelítési módot kínál.



A ChessBrain-rendszer felépítése

1. lista A ChessBrain XML-válaszcsomag

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env=
  http://www.w3.org/2001/12/soap-envelope
  xmlns:enc=
    "http://www.w3.org/2001/12/soap-encoding">
  <env:Body>
    <cbs:CBSGetPosResponse

xmlns:cbs="http://www.chessbrain.net">
      <return>
        rn1qk2r/p2p1ppp/bb2pn2/1p6/1P6/
        P2Q1NP1/1BP1PP1P/RN2KB1R b KQkq -
      </return>
    </cbs:CBSGetPosResponse>
  </env:Body>
</env:Envelope>
```

2. lista Egy XML kiszolgáló-állapotjelentés

```
<?xml version="1.0" ?>
<env:Envelope
xmlns:env="http://www.w3.org/2001/12/
                ➥soap-envelope"
xmlns:enc="http://www.w3.org/2001/12/
                ➥soap-encoding">
  <env:Body>
    <cbs:CBSSysInfoResponse
      xmlns:cbs="http://www.chessbrain.net">
    <Uptime days="1" hours="14"
            minutes="43" seconds="18" />
    <System proccnt="546" totmem="250.13"
            freemem="4.38"
            memu="98"
            cpustates="3627078,0,2151891,
            8160852"
            loadavg="0.50,0.30,0.33" />
      <Recv Bytes="2301480887.000000"
            Packets="16652816.000000"
            Errors="0.000000"
            Drop="0.000000" />
      <Send Bytes="2443488824.000000"
            Packets="12142245.000000"
            Errors="0.000000"
            Drop="0.000000" />
    </cbs:CBSSysInfoResponse>
  </env:Body>
</env:Envelope>
```

Botok, jelenlét és önálló játék

A SuperNode-kiszolgálónak van egy Shannon nevű botja (szálként megvalósítva), ami kapcsolódik az internetes játékkiszolgálókhoz és fönntartja a ChessBrain jelenlétét. A játékkiszolgáló felhasználói parancsokat írhatnak be, amikkel a ChessBraint kihívják, vagy figyelemmel kísérhetik a játékot. Jó szórakozás volt a Shannon programozása, és a bot ma már sokféle parancsot ismer. Nagy lehetőségek rejlenek a felhasználói bevitel helyett alkalmazott hálózati botokban. A ChessBrain fejlesztése során letöltöttem a Free Internet Chess Server (FICS) forráskódját, és lefordítottam egy Linuxot futtató régi Pentium 200 MMX Toshiba laptopon. A FICS C nyelvű, és a GCC gond nélkül fordította le. A kiszolgáló lehetővé teszi, hogy a felhasználók `telnet`-tel kapcsolódjanak az 50000-es kapura, és felhasználói név és jelszó segítségével bejelentkezzenek. Néhány hónap múlva a forgalom növekedése miatt a FICS kiszolgálót egy másik, a ➲ http://distributedchess.net tartományon elhelyezett ChessBrain-kiszolgálóra vittük át.

A felhasználóknak már több lehetőségük nyílik a játszmák követésére. Bejelentkezhetnek közvetlenül a `telnet`-tel a játékkiszolgálóra, ahol a ChessBrain játszik, vagy használhatják valamelyik játszmafigyelő programunkat.

Miután a ChessBrain képessé vált arra, hogy játékkiszolgálókon játszmákat folytasson, írtam egy Java-játszmafigyelőt, valamint PHP és Macromedia Flash alapú figyelőket is (➲ http://www.chessbrain.net/viewers.html). ***Anthony Bravo*** írt egy Java nyelvű hálózati figyelőprogramot, ami a világszerte éppen működő PeerNode-okat mutatja. A felhasználók a csomópontokra kattintva megtudhatják, hány aktív ügyfél van az adott országban.

A ChessBrain-honlapon található összes játszmakövető program a SOAP segítségével tart kapcsolatot a SuperNode-dal. Biztonsági megfontolásból az olyan bővítmények, mint a Java és a Macromedia Flash ActionScript nem engedélyezik, hogy a program más kiszolgálóhoz csatlakozzon, mint amelyikről letöltötték. Ezt az akadályt egy egyszerű XML proxy parancsállománnyal kerültem meg, ami HTTP GET-kéréseket fogad az egyik kiszolgálón, és az ügyfél nevében csatlakozik a SuperNode-kiszolgálóra. Ha például le akarjuk kérni a pillanatnyi játékállást a SuperNode-kiszolgálótól, a következőt írjuk a böngésző címsorába:

```
http://www.chessbrain.net/
➥xmlproxy.php?command=CBSGetPos
```

A kiszolgáló válaszul egy SOAP-csomagot küld, amilyet a fenti *ábrán* is láthatunk. A Mozilla böngészőben az oldal forráskódját megnézve látható a SOAP állomány.

A SuperNode megfigyelése

Egy kiszolgáló egészségi állapotának nyomon követése fontos része a rendszerfelügyeletnek. A fejlesztők szerencséjére a Linux sokféle módon megkönnyíti a kiszolgálók megfigyelését. A Linux */proc* virtuális fájlrendszere valóságos aranybánya az értékes rendszeradatok szempontjából, ami a fejlesztők számára könnyebbé teszi a rendszerműködés beállítását és nyomon követését. A */proc/net/dev* olyan adatokat mutat, mint a hálózati illesztőn küldött és fogadott bájtok és csomagok száma, a */proc/meminfo* pedig bőséges statisztikával szolgál a memóriahasználatról. Aki az adatokat nem szereti a */proc* fájlrendszerből kibányászni, a `sysinfo()` segítségével könnyen és gyorsan tölthet fel adatszerkezeteket olyan rendszerstatisztikákkal, mint a rendszerterhelés, a szabad memóriaméret és a futó folyamatok száma.

A SuperNode-kiszolgálótól a 2. *listán* látható példához hasonló rendszeradatok kérhetők le egy SOAP-hívással. A ChessBrain-közösség egyik tagja, ***Greg Davis*** írta az első programot a SuperNode megfigyelésére Perl nyelven, ami a SOAP-kérésre kapott választ a `top` parancshoz hasonló formában jeleníti meg.

A PeerNode megfigyelése

Mivel a ChessBrain-közösség számos tagja PeerNode-ügyfelet futtat sok gépen, egyszerű módon szerették volna nyomon követni egy gépcsoport állapotát. A PeerNode-ügyfelet úgy módosítottuk, hogy a 3434 kapura SOAP-kéréseket küldjön, így az ezen a kapun figyelő programok valós idejű állapotjelentést jeleníthetnek meg. Az első PeerNode-megfigyelőprogramot én írtam, aztán mások is elküldték saját változatukat. Grag Davis és ***Oliver Otte*** egy-egy Perl alapú programot tett közzé.

A legnépszerűbb PeerNode-megfigyelőt, a CBMoc programot ***Kris Drent*** írta Java nyelven.

Hordozható grafika és adatmegjelenítés

A ChessBrain rengeteg adatot gyűjt össze, és jelenleg is dolgozunk azon, hogy az adatok megjelenítési módját hasznosítsuk a rendszerfigyelés és -elemzés során. 3D-s navigációs eszközöket fejlesztünk, amik lehetővé teszik a hálózati játékmenet nyomon követését. ***Sven Herrmann***, házi 3D-szakértőnk OpenGL alapú leképezőt hozott létre, amit a SuperNode-megfigyelő programunkban használunk. Ezt a leképezőt használja majd a képernyővédőnk és a játszmakövető programjaink következő nemzedéke is.

Összegzés

Jelenleg a ChessBrain egy működő prototípus, ami több száz gépen futtatva sakkozik Linux, FreeBSD, Mac OS X és Microsoft Windows operációs rendszeren. Nemzetközimesterszinten játszik, és számos lehetőséget látunk a további fejlesztésére.

A ChessBrain jó példa arra, hogyan oldható meg egy összetett feladat nyílt forrású eszközökkel. Arra készülünk, hogy a ChessBrain kódját is nyílt forrású modellben fejlesszük tovább, azt remélve, hogy mások is csatlakoznak erőfeszítéseinkhez.

A ChessBrainben az a csodálatos, hogy annyira sokféleképpen lehet hozzájárulni. Bárki részt vehet benne, akinek van egy gépe és internet-hozzáférése. Elég letölteni egy PeerNode-ügyfelet a ChessBrain-honlapról, futtatni egy vagy több gépen, és a ChessBrain máris nagyobb lett.

Arra törekszünk, hogy a „legtöbb, egyetlen játékot játszó számítógép” hivatalosan elismert világrekordját állítsunk fel. Már kapcsolatba léptünk a világrekordokat nyilvántartó londoni irodával és több hivatalos sakkszövetséggel is.

A ChessBrain mögött erős csapat áll, többek között ***Peter Wilson***, a sakk világszövetség (FIDE) számítógépes sakk- és internetbizottságának korábbi elnöke. Pillanatnyilag megfelelő helyszínt keresünk egy nyilvános és internet alapú bemutatóhoz, ami magában foglalná az osztott számításhoz és sakkhoz kapcsolódó Guinness világrekord felállítását is. Érdemes figyelni a ChessBrain-honlapon a hivatalos bejelentést.

Köszönetnyilvánítás

Szeretném megköszönni ***Janus Daniels***, ***Cedric Griss*** és a ChessBrain-közösség más tagjainak a támogatását, amit a cikk írásához nyújtottak. Az elérhetőségek és a ChessBrain-kikicsoda megtalálható a
➲ http://www.chessbrain.net/friends.html weboldalon.

Linux Journal 2003. szeptember, 113. szám

Carlos Justiniano (cjus@chessbrain.net)
A programipar húszéves tapasztalattal rendelkező veteránja. A ChessBrainben számos jelmondatát átülteti a gyakorlatba: „Fogadd el a bonyolultságot!”; „Ne kézenfekvő feladatokat keress!” és „Oldd meg nyílt forráskóddal!”.

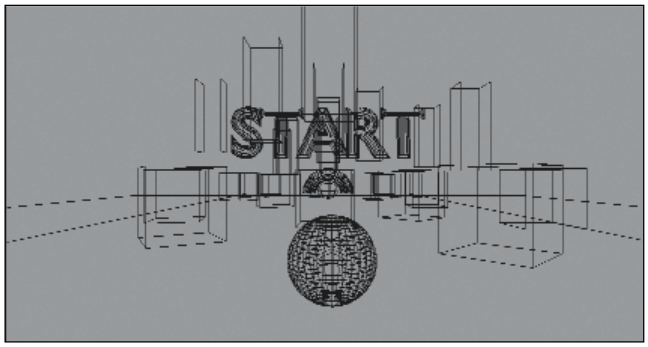
Játékmotor a Blenderben (1. rész)

Ismét a Blender képességeit fogom ismertetni, az elmúlt egy hónapban azonban alkalmam volt jobban átgondolni a tanmenetet. Ezért ebben a részben a játékmotorban használható érzékelőket (Sensors) és végrehajtó szerveket egyesével mutatom be, megalapozva a további játékkészítést.
Folytassuk a tanulást az elméleti anyag elsajátításával – a szükséges ismeretek birtokában már könnyebben érthető lesz játékunk elkészítésének a módja is. Mivel azonban a tárgyakhoz rendelhető változókat már az előző részben megismertük, folytassuk a sort a különféle érzékelőkkel!
A Blender játékmotorja tizenegyféle érzékelőt kínál, ezekből természetesen tetszőleges számút rendelhetünk egy-egy tárgyhoz. Az első ilyen az `Always` nevet viseli, működése szerint a kimenetén meghatározott időközönként jelzést ad. Minden érzékelőnek három olyan alapvető tulajdonsága van, amelyekkel beállíthatjuk a kimenet viselkedését. Az egyik ilyen tulajdonság az ***Active***, amelyet az érzékelő neve alatt található két kapcsolóval választhatunk ki. Értéke `True` vagy `False` lehet. Ezzel határozhatjuk meg, hogy amikor az érzékelő kimenete az alapállapottól eltérő lesz, annak logikai értéke igaz vagy hamis legyen. Fontossága abban rejlik, hogy minden érzékelő kimenete egy feltételt vizsgáló csomópont bemenetéhez kapcsolódik, ahol a feltétel egy logikai vizsgálat is lehet. A másik jellemző tulajdonság a – ***freq:***, vagyis az érzékelő működési frekvenciája. Ezzel határozhatjuk meg, hogy a játékmotor milyen gyakran vizsgálja az adott csomópont feltételeinek a teljesülését. Ezt az értéket 1/50-ed másodpercben mérve kell megadni, és a beállítására szolgáló mező az érzékelő neve alatt középen található. A harmadik alapvető tulajdonság pedig a név alatt jobbra található ***Inv***, amellyel meghatározhatjuk, hogy a kimenet állapota a feltételek teljesülésekor vagy azok nem teljesülésekor váltson-e aktív jelszintre.
A `Keyboard` érzékelővel, miként a neve is mutatja, a billentyűzet eseményeit figyelhetjük – az alábbi beállításokat figyelembe véve. A ***Key*** az elsődleges billentyű, amelyre az érzékelő válaszolni fog. A ***Hold*** felirat mellett található két mezőben adhatjuk meg azt a két módosító billentyűt, ami a jelzés kiváltásához elengedhetetlen. Természetesen amennyiben nem bonyolult billentyűkombinációkban gondolkodunk, ezek megadása nem kötelező. Ezeket a mezőket könnyedén beállíthatjuk, ha az egérrel a mezőre kattintunk, ezután pedig a ***Press a key*** felirat megjelenésekor lenyomjuk a kívánt billentyűt. Módosító billentyű lehet a bal és jobb oldali Ctrl, az Alt, a Shift és természetesen bármelyik másik billentyű is, de célszerű a fentiek közül választani, mert ezek eredeti működése is a többi billentyű működési módjának a megváltoztatása. A ***LogToggle*** felirat melletti mezőben beállíthatjuk, hogy a megadott billentyű lenyomásakor valamilyen szöveges üzenetet is átadjunk a ***Target*** mezőben megadott tárgynak.
A `Mouse` érzékelővel az egér eseményeire válaszolhatunk a játékban. Kevés beállítási lehetősége közül az első három az egérgombokra vonatkozik, míg a ***Movement*** az egér mozgatásakor hozza működésbe az érzékelő kimenetét, a ***Mouse Over*** pedig akkor, ha az egérmutató a tárgy fölé kerül. Ezt az utóbbit csak abban az esetben tudjuk majd kipróbálni, amikor a Blenderrel futtatható állományt készítünk a játékból.
A `Collision` típusú érzékelő alkalmas rá, hogy egy meghatározott tulajdonsággal vagy anyaggal rendelkező tárggyal való érintkezést észleljünk a játékban. Az ***M/P*** kapcsolóval állíthatjuk be, hogy tulajdonságot vagy mintázatot szeretnénk-e figyelni. Az érzékelő akkor fog jelzést adni, amikor a kijelölt tárgy azzal a meghatározott tulajdonsággal vagy anyaggal rendelkező tárggyal érintkezik, amelyet itt beállítottunk.
A `Touch` érzékelő működése csak annyiban tér el az előzőtől, hogy nem adhatunk meg tulajdonságot, csakis a tárgyak anyagát figyeltethetjük vele.
A `Near` érzékelővel akkor adhatunk jelzést, ha a tárgyunk egy adott tulajdonsággal rendelkező másik tárgy közelébe ér. A `Dist` értékkel a bekapcsoláshoz szükséges távolságot adjuk meg, míg a `Reset` értékkel azt a távolságot, amelynél ha a két tárgy messzebb kerül egymástól, az érzékelő kimenete passzívvá válik.
A `Radar` típusú érzékelő működése megegyezik a kinti világban is használatos rádiólokátorokéval, vagyis egy kúp alakú csóván belül lévő tárgyak érzékelésére szolgál. Itt is megadható, hogy milyen tulajdonsággal rendelkező tárgyakat keressen, a `X`, `Y`, `Z` kapcsolókkal pedig megadhatjuk, hogy a forrásként szolgáló tárgynak (amelyikhez az érzékelőt hozzárendeljük) melyik tengelye alkossa a kúp tengelyét. Az érzékelő kúpjának a csúcsa egybeesik a tárgy érvényben lévő középpontjával. Az `Ang` értékkel adhatjuk meg a kúp nyílásszögét, míg a `Dist` értékkel a hatótávolságot.
A `Property` érzékelő alkalmas arra, hogy figyeljük a tárgy egy adott tulajdonságát. A beállításoktól függően jelzést kaphatunk, ha a ***Prop*** mezőben beállított tulajdonság egyenlő (`Equal`) vagy nem egyenlő (`Not Equal`) a ***Value*** mezőben beállított értékkel, vagy akkor is, ha az érték megváltozott (`Changed`), esetleg egy adott tartományon belül van (`Interval`). A működést kiváltó feltételt a legnyíló listából választhatjuk ki, az előbbieknek megfelelően. Amennyiben az `Interval` feltételt választjuk, a lista alatt két mezőt találhatunk, amelyekben a tartomány alsó és felső határát adjuk meg.
A `Random` érzékelő működése minden jel szerint megegyezik az `Always` típusúéval, semmilyen szemmel látható különbséget nem fedeztem fel.
A `Ray` érzékelő hasonlít a korábban bemutatott `Radar`-hoz, itt azonban az érzékeny terület nem kúp alakú, hanem egyenes vonal. Ennél is beállítható, hogy az érzékelő állapota anyagtulajdonságra vagy játékmotorbeli tulajdonságra változzon-e, továbbá a `Range` értékkel a hatótávolságot is meghatározhatjuk. A lenyíló listában szereplő tengelyek szabják meg, hogy az érzékelő sugár a kiindulási tárgynak melyik tengelyével párhuzamosan haladjon, és ebben az esetben természetesen az előjel határozza meg a kiindulási irányt. Ez az érzékelő alkalmas például annak megállapítására is, hogy egy lézersugár eltalált-e egy adott tárgyat vagy a főhős belépett-e az infravörös kapu sugarába.
A `Message` típusú érzékelő a sorban az utolsó, egy üzenet elfogására szolgál. A ***Subject*** mezőben határozhatjuk meg azt a szöveget, ami a tárgyhoz érkező üzenetek azonos nevű

mezőjében szerepelhet. Amennyiben megadunk egy ilyen értéket, a tárgyhoz rendelt érzékelő csak ilyen üzenetek érkezésekor reagál majd.
Az előző részben ismertettem már a vezérlők (***Controllers***) típusait, így ennek ismétlésétől eltekintve a hatásokat kiváltó csomópontok (***Actuators***) tárgyalására térek át.

1. kép Hangállomány betöltése

Az első ilyen hatás a `Movement`, amivel a tárgy mozgását tudjuk befolyásolni. Minden értéket a három térbeli tengelyre vonatkoztatva háromdimenziós vektorként adhatunk meg.
A `Force` a tárgy mozgási energiáját határozza meg, a `Torque` pedig a forgatónyomatékát. A `dLoc` a tárgy helyzetének változtatására szolgál, míg a `dRot` a tárgy elfordulását változtatja meg. A `linV` a tárgy haladási sebességére vonatkozik, és ide tartozik még a három koordináta után található ***add*** kapcsoló – ennek alkalmazásával a tárgyat a megadott értékeknek megfelelően gyorsíthatjuk vagy lassíthatjuk. Ha ezt a kapcsolót használjuk, akkor a beállított értékek hozzáadódnak az érvényben lévő sebességhez. Az `angV` vektorral határozhatjuk meg a tárgy szögsebességét. A fent felsorolt vektorokra vonatkozóan beállíthatjuk, hogy a Blender az értékeket a tárgy helyi koordinátarendszerében vagy a jelenet koordinátarendszerében értelmezze-e. Amennyiben a tárgy helyi koordinátarendszerét szeretnénk használni, akkor az adott vektor sorának végén lévő ***L*** kapcsolót is be kell kapcsolni.
A `Constraint` hatás alkalmazásával elérhetjük, hogy tárgyunk mozgása valamely tengely mentén korlátozott legyen. A kívánt tengelyt a legördülő listából választhatjuk ki, majd a `Min` és a `Max` értékekkel adhatjuk meg a korlátozás mértékét.
A `Camera` hatás alkalmazásával a kamerának adhatunk utasítást arra, hogy a ***OB:*** mezőben megadott tárgy felé forduljon. Itt is használhatunk különféle megkötéseket. Az egyik ilyen a ***Height***, amellyel a kamera magassága állítható be, a másik pedig az ***X*** és ***Y*** kapcsolók egyike, ezzel azt érhetjük el, hogy a kamera a céltárgy kiválasztott tengelye mögött próbáljon maradni. Természetesen előfordulhat, hogy a kamera nem tud megfelelően a tárgy felé fordulni, például akkor sem, ha az X tengelyt beállítjuk, és a kamera már alaphelyzetben is a céltárgy X tengelyének meghosszabbításán helyezkedik el.
A `Sound` meglepő módon hangállomány lejátszására alkalmas. Amikor azonban ezt a hatást választjuk ki, alaphelyzetben hangfájl nem található a legördülő listában. Hogy itt is megjelenjen valamilyen választási lehetőség, először a nézetek váltására szolgáló gombsoron ki kell választanunk a hangszóró ikonnal jelzettet, és az anyagtulajdonságok hozzáadásához hasonlóan a nézet fejlécének segítségével be kell töltenünk egy hangállományt. Ennek megértését szolgálja az *1. kép* is, amelyen piros és zöld keretek jelölik a megfelelő gombokat.
A Blender képes WAV-állományok betöltésére és lejátszására.
A `Property` hatást (effect) alkalmazhatjuk olyan esetekben, amikor a tárgy valamely játékmotorbeli tulajdonságát szeretnénk megváltoztatni. A ***Prop*** mezőben megadott változó értékét a legördülő listából kiválasztott ***Assign*** menüponttal változtathatjuk meg, vagy hozzáadhatunk egy előre meghatározott értéket, feltéve, hogy a listából a ***Add*** pontot választjuk ki. Amennyiben ebből a listából a ***Copy***-t választjuk, akkor a létrehozott tárgy örökli az előző tárgy tulajdonságait. Ebben az esetben a ***OB:*** mező tartalmazza a tulajdonság forrását, emellett a ***Prop*** határozza meg a forrástulajdonságot, míg a cél a korábban beállított ***Prop*** értéke lesz.
Az ismertetést a könnyebben érthető hatásokkal folytatom: a `Message` alkalmas arra, hogy egy adott tárgynak



2. kép A játék kezdete

(ezt határozza meg a ***To*** mező értéke) – vagy amennyiben nem adunk meg céltárgyat, akkor minden tárgynak – üzenetet küldjünk. A ***Subject***-ben megadhatjuk az üzenet tárgyát, ez alapján a `Message` érzékelő el tudja dönteni, hogy kell-e foglalkoznia a beérkező üzenettel. A ***T/P*** kapcsolóval választhatunk szöveges üzenet vagy egy tulajdonság értékének az elküldése között.
A `CD` hatással zenei CD-t tudunk lejátszani a játékban. Lejátszhatjuk az egész lemezt, egy kiválasztott számot, a hatás segítségével pedig módosíthatjuk a lejátszás hangerejét, megállíthatjuk, majd újra elindíthatjuk a lejátszást.
A `Game` alkalmas a játék viselkedésének a megadására. Segítségével kiléphetünk a játékból, újraindíthatjuk azt, vagy egy másik játékot tölthetünk be. Amikor egy másik játék betöltése mellett döntünk, természetesen a játékot tartalmazó fájlnevet is meg kell adni.
És végül a `Visibility` hatás segítségével a tárgy láthatóságát szabályozhatjuk, például eltüntethetjük a már kilőtt és felrobbant űrhajót.
Nos, ebben a hónapban megtárgyaltuk a játékmotor legfontosabb összetevőit, ami ugyan egy kissé száraz és nagyon is elméleti anyag volt, ám a következő részben befejezzük a játékot (a működés szempontjából nézve), és a most kimaradt hatásokkal is megismerkedünk. Mint azt korábban már említettem, játékunk célja az lesz, hogy egy gömbszerű főhőssel anélkül jussunk át az akadálypályán, hogy a kalapácsok összelapítanának minket vagy leesnénk a pálya szélén. A játékot nehezíti az is, hogy a főhős a nem mozgó akadályokkal ütközve pattog, így könnyen előfordulhat, hogy egy ütközést követően pont egy kalapács alá kerül. A külső kinézetre vonatkozóan mellékeltem egy képet (*2. kép*). A távolban látható a kijárat boltíve, míg a főhős a nézőponthoz legközelebb eső gömbszerű alak.
A decemberi viszontlátásig mindenkinek kellemes ismerkedést és 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.

Műsorszórás az interneten

Ismerkedjünk meg a digitális műsorszórás rejtelmeivel és a kiszolgálóalkalmazásokkal!

Bizonyára sokan hallottak már a címben foglalt dologról, ám kevés olyan embert ismerek, aki részletesen ismerné a folyamat mikéntjét, tisztában lenne a szolgáltatás alapjaival, vagy épp rendszeresen igénybe venné azt. Jelen cikk egyik célja épp ezért az, hogy megismertesse az internetes műsorszórás alapvető technikai jellemzőit, vázolja a működését és eloszlassa az esetleges tévhiteket, amelyek a témával kapcsolatban a hiányos adatok miatt kialakultak. A cikk fennmaradó részében pedig a teljesség igénye nélkül szeretném néhány fő jellemzőjével bemutati a témához kapcsolódó linuxos programokat, megoldásokat. Az egyes programok részletes bemutatása a sorozat későbbi részeiben történik majd. Vágjunk bele!

Mi is ez tulajdonképpen?

Valószínűleg mindenki számára természetes és megszokott jelenség a 21. században a televíziózás vagy a rádiózás, arra azonban már kevesebben gondolnak, hogy mi történne, ha ezt a tartalmat egy teljesen másfajta közvetítő közeg juttatná el a közönséghez. Ilyen közeg lehet például az internet, amely tulajdonságait tekintve minden szempontból megfelel: emberek milliói használják, széles körben elterjedt, az előrejelzéseket figyelembe véve szinte minden területet meg fog hódítani; alkalmas adatok továbbítására, a végpontokon elhelyezkedő számítógépek képesek a digitális kép- és hangtartalom lejátszására stb. Egyszóval olyan környezetről van szó, amely minden adottsággal rendelkezik, csupán ki kell aknázni ezeket. A helyzet természetesen mégsem egészen ilyen egyszerű. Bár a fent leírt dolgok mindegyike igaz, van néhány – az átlagfelhasználók számára láthatatlan – technikai akadály, amelyek áthidalása jelenleg is folyamatban van. Ezek az akadályok miatt jár még csak gyerekcipőben a Világhálón történő műsorszolgáltatás.

A jövő

Hogy miért igyekeznek mégis elterjeszteni a technológiát? A válasz egyszerű: az új eljárás számtalan előnnyel jár. Egyik legfőbb előnye például az, hogy a szolgáltatást használó személy beavatkozhat a műsorfolyamba, vagyis kétirányú adattovábbítás van jelen a

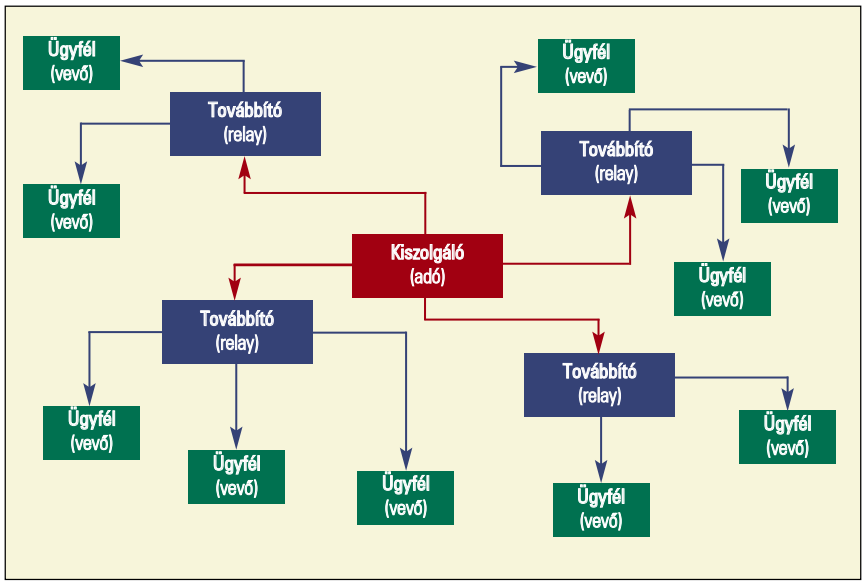
kapcsolat során – szemben a televízió- és rádióadással, ahol csak a stúdióból a vevő felé áramlik az adat. Nem kell hozzá túl sok fantázia, hogy felmérjük, miféle új szolgáltatások bevezetését teszi lehetővé egy ilyen technikai adottság: végre egyszerűen megvalósulhat az interaktív tévéadás, ahol a néző maga is alakíthatja a műsor menetét, a film tartalmának alakulását, vagy épp egy focimeccs közvetítése során megválaszthatja a kameraállást, bármikor visszanézheti az ismétléseket stb. (Ezzel egyébként most is kísérleteznek a hagyományos lehetőségek mellett: az adás megszokott módon jut el a tévénézőkhöz, de egy visszirányú kapcsolatot is biztosítanak ez lehet telefonos vagy műholdas), amelyen keresztül a felhasználó elküldheti az igényeit a „központnak”.
A másik fontos előny abból adódik, hogy a vevő szinte közvetlen kapcsolatban áll az adóval, így a nézettség egyszerűen mérhetővé válik; azonnal megoldható, hogy csak az nézhessen egy műsort, aki előfizetett rá. Ha valaki mondjuk nincs előfizetve, de most azonnal szeretné élvezni az adott adást, akkor arra bankkártyával azonnal előfizethet, és máris szemmel követheti – nem kell a kábeltévéseket megvárni, hogy mindezt lehetővé váljon. Nem kizárt annak megoldása sem, hogy a DVD-filmekhez megszokott módon feliratot, illetve nyelvet válasszunk ahhoz a műsorhoz, amit épp nézünk. Ezenkívül olyan információhoz is hozzájuthatunk, amelyet tévénézés vagy rádióhallgatás közben egyébként nem érhetünk el, például az adott műsor címéhez, a hozzá kapcsolódó összes adathoz, a következő programhoz. A legnagyobb előnye mégis az, hogy mindehhez nincs szükség új berendezésre, új eszközökre, új adatkapcsolatra, antennára – a jelenlegi, internetelérésssel rendelkező számítógép is megteszi.

A jelen

Meg kell jegyeznem, hogy azért az itt leírt előnyök kiaknázása még nem történt meg. Mint már említettem, még csak a gyerekcipős korszakban járunk, és jelenleg a hálózatok jelentős része sem teszi lehetővé az ehhez szükséges sávszélességet, néhol protokollt, de ennek ellenére szegényesebb szolgáltatási színvonallal már most is számtalan helyen alkalmazzák. Hogy tiszta vizet öntsünk a pohárba, szeretnék néhány szót szólni a folyamatról, a technikai nehézségekről, hogy ezek miből fakadnak, és hogy ezeknek a megoldására milyen kilátások vannak.
Maga a műsorszórás folyamata úgy zajlik, hogy van egy kiszolgáló, amely valamelyik előre meghatározott hálózati kapujára egy adatfolyamot továbbít.
A kiszolgáló ezen kapujára kapcsolódva az ügyfelek hozzáférhetnek ehhez az adatfolyamhoz. Az egész úgy működik, mintha csak letöltenénk egy weblapról, de az adatfolyam itt nem kerül mentésre,

hanem egy átmeneti tárba jut, ahonnan egy – az adatfolyamot lejátszani képes – program kiolvassa és megjeleníti. A műsorfolyamnak tehát olyannak kell lennie, hogy bármelyik pillanatától olvasva hiba nélkül lejátszható legyen, tehát a folyam jellemzői valamilyen ismétlődő módon rövid időnként elérhetőek kell legyenek. Az ilyen adatfolyamokat szakaszolhatónak (streamelhetőnek) nevezzük, ilyen például az MPEG layer 3, az MPEG1, az MPEG2, a Winows Media Video/Audio vagy a RealMedia formátum. Ezek közül az utóbbi kettőt csak hang, vagy kép és hang továbbítására a felhasználó számára átlátszó módon egyaránt alkalmazzák, ezeket a formátumokat már kifejezetten az internetes műsorszórásra találták ki (ami természetesen nem jelenti azt, hogy helyi adattárolásra nem alkalmasak).

A fentiek lejátszásához magától értetődően sávszélességre van szükség mind az adó, mind a vevő oldalán. A sávszélesség az adás minőségének függvénye – az egyik technikai akadály rögtön ebből adódik. A kiszolgálók általában megfelelően nagy sebességű kapcsolattal rendelkeznek, ám ha sokan kapcsolódnak hozzá, pillanatok alatt ki lehet meríteni. Gondoljunk csak bele: gyenge minőségű képanyag továbbításához mostanában már elég 384 Kbit (egy szabvány ADSL kapcsolat sebessége – nem véletlen, hogy épp ilyen méretűre szabják az adatfolyamot). Ha egy kiszolgáló a hazai viszonyok mellett 100 Mbit sávszélességgel rendelkezik, az mindössze 260 ügyfél kiszolgálását teszi lehetővé, ami igazából semmi a televíziók több milliós nézőtáborához képest, és hosszú távon mindenképp halálra van ítélve. Az internetes rádiókkal is ugyanez a helyzet, bár azok kevesebb sávszélességet igényelnek (kb. a harmadát, a felét), tehát nem oldódik meg, hogy felhasználók százezrei kapcsolódhassanak az adáshoz. Ezt most úgy próbálják kiküszöbölni, hogy több szolgáltatónál több kiszolgálót üzemeltetnek, a nagyobbak gigabit nagyságrendű kapcsolattal, és a kapcsolatokat elosztják az egyes gépek között. Mondanom sem kell, hogy ez milyen költséges, és a közönség növekedésével milyen korlátokba ütközhet, tehát csak egy „átmeneti" megoldásról van szó. (Ma egyébként szinte az összes ilyen internetes rádióállomás így működik.) A viszonylagosan kicsi közönség ugyanakkor nehézkessé teszi egy ilyen állomás megélhetését, nem jelent ugyanis piacot a reklámipar számára, ám az adás költségei, ha nem is olyan nagyok, ugyanúgy jelen vannak (a műsor jogdíjai, a fájlformátum jogdíja, a sávszélesség költsége stb.), s ezeket meg kell fizetni.



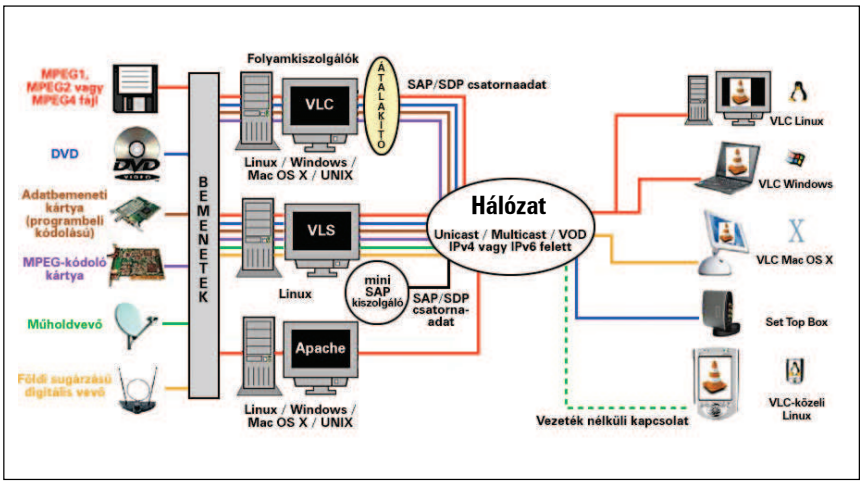
1. ábra A kapcsolatok növekedésének egyik módja

Egy lehetséges megoldás

Erre a nehézségre azonban már megszületett a megoldás, már csak azt kell megvárni, hogy elterjedjen. A megoldás maga a hálózatok felépítésében rejlik, pontosabban az adattovábbítás módjában. Az interneten használatos TCP/IP protokoll IP-rétege felépítéséből adódóan csak az „unicast" (egyéni) módú csomagküldést teszi lehetővé a megszokott módszerrel, az egész hálózatra nézve. Ez a gyakorlatban azt jelenti, hogy ha egy kiszolgálótól két ügyfél ugyanazt az adatfolyamot kéri (tehát ugyanabban az időben két gép is ugyanazt tölti le ugyanabban az időpillanatban), akkor a kiszolgáló mindkét gépnek kétszer elküldi ugyanazt a csomagot, ami tőle kétszer annyi kimenő sávot foglal le, s mivel ezek az adatfolyamok meglehetősen nagyméretűek, ez okozza a jelentős sávszélesség-igényt. Természetesen ez nemcsak az adó és a vevő közötti útvonalon áll fenn, sok adó esetén a teljes gerinchálózat is eldugulhat tőle. Létezik már azonban az alkalmazott IP protokoll kiterjesztése, amely már támogatja a „multicast" (csoportos) módú csomagküldést, s ez egy csapásra megoldja a felesleges sávfoglalást. Amikor ugyanis egy kiszolgálótól két ügyfél is ugyanazt a csomagot kéri, akkor a kiszolgáló mindössze egy üzenetet küld, amelyben a csomag összes címzettje fel van sorolva, és a csomag útja során csak a megfelelő csomópontban sokszorozódik, ott, ahol a csomag útja kettéválik. Így ha a fa felépítésű hálózatban haladunk, minden egyes szakaszon csak egy darab ugyanolyan csomag van jelen. Ezt azonban a hálózati eszközöknek, csomópontoknak is támogatniuk kell, ezért meg kell várni, amíg szép fokozatosan elterjed. A jelenlegi állapot szerint a csoportos üzenetküldés ugyan létezik, szabványok vonatkoznak rá, ám a használatához különleges beállításokra van szükség, az adónak előre meghatározott ügyfélcsoporttal kell rendelkeznie, és ami a legnagyobb probléma: nem tudjuk bárhol igénybe venni, ugyanis csak kis „szigetek" formájában létezik. Ez azt jelenti, hogy az internet egyes részein vannak meg csak a technikai feltételei, nincs teljes – de még nagy részt magába foglaló – lefedettség sem. Régóta törekszenek csoportos üzenetküldést támogató gerinchálózat megteremtésére, nem is sikertelenül: üzemel egy MBONE (Multicast Backbone) nevű virtuális gerinchálózat, ám ehhez többek között még Magyarországnak sincs kapcsolódási pontja (nálunk is csak kísérleti jelleggel léteznek fent említett „multicast-szigetek").

Emellett természetesen a végpontok jelentős részének is gyorsulnia kell körülbelül egy nagyságrendnyit, hogy az elején említett minőségű és szintű műsorszolgáltatás jöhessen létre.

Most, hogy ilyen remekül leírtam, hogy igazából nem is működik, itt az ideje, hogy bemutassam: valójában mégiscsak üzemel, s ezrek használják.



2. ábra Összetett lehetőségek a VideoLAN segítségével

Remekül működött például az elmúlt időszak során lezajlott valóságshow-áradat közepette, amikor is a helyszíneken zajló eseményeket a nap 24 órájában figyelemmel kísérhettük az interneten keresztül, csupán egy médialejátszó segítségével. A legtöbb helyen egy gyenge és egy jobb minőségű adatfolyam egyaránt elérhető volt (valójában ez is elég gyenge volt, de a célnak tökéletesen megfelelt), hogy kielégítsék a különböző sávszélességű kapcsolattal rendelkezők igényeit. Ez egyébként minden nagyobb ilyen szolgáltatás háza táján így működik. Ezenkívül számtalan rádióállomás működik folyamatosan, mp3-, asf- vagy vma-adatfolyamot „sugározva". Igazából ezek a népszerűbbek, hiszen nem nagy méretűek, és a minőségük megközelíti, vagy éppen meghaladja a földi sugárzású rádiók hangminőségét (beleszámítva az apró légköri zajokat). Ami a népszerűség mellett szólhat még, az az, hogy a lejátszóprogramokkal egyszerűen lehetőségünk nyílik az adatfolyam mentésére, tehát változatlan minőségben rögzíthetjük a meghallgatott adást (részben vagy egészben). A rádiók hallgatásához minőségtől függően elegendő a 80–500 kbit/s-os sávszélesség, de egy igen jó minőségű adafolyam lejátszásához már 130–160Kbit is megteszi, amit egy mai ADSL-es vagy kábelmodemes csatlakozás is bőségesen teljesít, és mellette még egyéb letöltések is folytathatók (webezés, levélolvasás stb.). Maga a szolgáltatás (kép, hang egységesen) a legtöbb helyen HTTP protokollon keresztül működik, amelyet minden internetre kötött gép feltétlenül ismer, hisz a világhálón elérhető legelterjedtebb szolgáltatásról van szó. Emiatt az ilyen módon elérhető adást sokan webrádiónak is nevezik. Most, hogy nagyjából ismerjük az effajta műsorszolgáltatás mikéntjét, szót ejthetünk arról, hogy milyen módszerekkel vehetjük igénybe ezt kedvenc operációs rendszerünk, a Linux segítségével. Rögtön bajban is vagyok, mivel nem tudom, hogy melyikkel is kezdjem, ugyanis igen jó támogatást nyújt mind a vevő, mind az adó oldalán. Remek lehetőségeket biztosít az adatfolyamok vételére és lejátszására, valamint az ilyen jellegű adások szórására, tehát a kiszolgálói feladatra. Azt hiszem, helyesebb volna először a fogadó (vevő) oldal programjait bemutatni, lássunk néhányat ezekből!

XMMS

Ez valójában a jól ismert MP3-lejátszó, amely a híres windowsos Winamp GPL-es változata. Számunkra most azért érdekes, mert kiválóan alkalmas internetes rádióadások hallgatására. Alapértelmezetten képes lejátszani az MP3 vagy OGG Vorbis alapú adatfolyamokat. Szerencsénkre számtalan olyan rádióadás létezik, amelyik ebben a formátumban sugároz (az ilyen rádióadók címei megtalálhatók például a ➲ http://www.shoutcast.com címen kategorizálva, kereshető formában). Csupán annyi a dolgunk, hogy a megszerzett címet URL-ként hozzáadjuk a lejátszási listához, majd elindítjuk azt. Léteznek azonban olyan rádióadók is, amelyek például asf formátumban sugároznak. Nem kell megijednünk, ugyanis egy XMMS bővítmény telepítésével képesek leszünk minden olyan formátum lejátszására, melyet az ***avifile*** könyvtár támogat. Ezek után már mindegy lesz számunkra, hogy épp melyik formátumú rádiót hallagtjuk.

RealOne Player

Ez a lejátszó alkalmas az egyik legősibb, szakaszolható formátum fogadására. Bizonyára sokan hallottak már a RealMedia által kifejlesztett, interneten sugározható formátumról, ugyanis ez volt az első, amelyik széles körben elterjedt. Az ilyen fájlokat a linuxos RealOne Player segítségével tudjuk egyszerűen lejátszani. Itt is hasonló elven kell meghallgatni a rádióadásokat, nevezetesen be kell írnunk egy RealMedia formátumban sugárzó kiszolgáló címét (ilyen címeket a ➲ http://www.real.com címen találhatunk). A lejátszó tudománya természetesen itt nem áll meg, ez ugyanis egy olyan formátum, amely kép továbbítására is szolgál, így tévéadások fogadására is lehetőségünk nyílik.

SHOUTcast

Ez egy kiszolgálóoldali adatfolyamszóró program, amely lehetővé teszi, hogy a hálózati csatlakozással rendelkező gépünkből rádióállomást készíthessünk. Maga a program úgy működik, hogy egy, a SHOUTcasttal együttműködni képes program egy előre kiválasztott bemenetről érkező hangot (vagy adatfolyamot) MP3 formátumúvá alakít (ilyen lehet például egy XMMS bővítmény), majd ezt „átküldi" a kiszolgálónak, amely – például HTTP protokollon keresztül – elérhetővé teszi a hozzánk csatlakozó számítógépek számára. Ezek után ha valaki csatlakozni szeretne a gépünkhöz mint rádióállomáshoz, nyugodtan megteheti, csak be kell írnia a lejátszóba gépünk IP-címét és a csatlakozási kapu számát, majd élvezheti a gépből kiáramló zenét. A programnak számtalan beállítási lehetősége van. Az egyik nagy csoportba a hálózati jellemzőkre vonatkozó kapcsolók sorolhatók, például a kimeneti kapu beállítása, a hálózati ügyfélazonosítás bekapcsolása, a lehetséges csatlakozások számának beállítása, és még sorolhatnám – ezt részletesen egy későbbi cikkben fogom bemutatni.

icecast

Ez egy GPL licenszes kiszolgálóprogram a SHOUTcast mintájára. Felhasználási szerződésén kívül egy fontos dologban különbözik az egyébként vele együttműködő SHOUTcasttól: az icecast téma keretein belül hozzáfejlesztett adatfolyam-készítő program a szabad Ogg Vorbis formátumot támogatja. Ennek az az előnye, hogy a SHOUTcast által előnyben részesített MP3 formátummal

szemben az Ogg Vorbist nem védik szabadalmak, bármilyen nemű használata nem ütközik törvénybe. Maga az icecast valójában három részből áll: először is a fent említett kiszolgálóból, amely a számára készített adatfolyamot a számítógépünkhöz kapcsolódó ügyfelek számára elérhetővé teszi. A másik része a ***shoutlib*** névre keresztelt függvénykönyvtár, amely a kiszolgálóval történő kapcsolattartást teszi lehetővé. A harmadik egység az a program, amelyik az ogg formátumú adatfolyamot előállítja a kiszolgálórész számára, a neve: iceS.

bitchasshola

Ez a program webalapú felület az icecast kiszolgálóhoz; lehetővé teszi, hogy különböző lejátszási listákat állítsunk össze, majd ezeket adatfolyamként elindíthatjuk, így a hozzánk csatlakozó ügyfelek rádióadásként hallgathatják.

CGIcast

A csomag néhány Perl-parancsfájl gyűjteménye, arra képes, hogy csupán egy webkiszolgáló program segítségével lehetőségünk nyíljon valós idejű internetes rádióállomás működtetésére.
A módszer akkor előnyös, ha valaki egy szolgáltatónál bérel webes tárhelyet, és ezen keresztül szeretne sugározni. Ebben az esetben nincs lehetőség a fentebb említett kiszolgálók telepítésére, viszont a legtöbb szolgáltatónál engedélyezett a CGI parancsfájlok futtatása. Három fő részből áll: az első a ügyfélgépen fut, ahonnan maga a műsor „készül". Innét továbbítja az MP3-szakaszokat a következő részhez, amely már a kiszolgálón fut, s tárolja az adatfolyam legutóbbi pár másodpercét. A program harmadik része fogadja a hallgatókat, és némi késleltetéssel elérhetővé teszi számukra a műsort.

Fluid Server Streaming

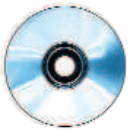
Ez egy Java nyelven írt GPL licenszes kiszolgálóprogram, amely MP3 formátumú hang szórására képes bármilyen TCP/IP alapú hálózaton. Szolgáltatásai között szerepel például, hogy képes egy Winamp/XMMS által használt lejátszásilista-formátum (m3u) adásához, ami számottevően egyszerűsíti a program használatát. Másik érdekes képessége, hogy az adatfolyamot továbbítani (relay) tudja egy másik Fluidot futtató kiszolgálónak, így kiszolgálók egész hálózata hozható létre, amely a bevezetésben is említett sávszélesség-igényt csökkenti azáltal, hogy a felhasználók nem csak egy kiszolgálóhoz csatlakozhatnak (*lásd az 1. ábrát*). Mindezt úgy oldja meg, hogy egy bemenő kapun fogadja az adatfolyamot, majd mindezt a kimeneti kapuján továbbítja. Ez azt is jelenti, hogy így bármilyen másik kiszolgálóval együtt tud működni, annak adatfolyamát továbbítani képes. Ezenkívül lehetővé teszi, hogy a kiszolgálóhoz `telnet` protokollon keresztül csatlakozva adatokat kérjünk le az állapotát illetően (csatlakozott felhasználók, terheltség stb). Ha még ez sem volna elég, a csomagban egy Java applet is megtalálható, amelyen keresztül egyéb érdekes dolgokat tudhatunk meg az éppen sugárzott adatfolyamról.

VideoLAN

A csomagban található kiszolgáló segítségével képanyag sugárzására is képesek vagyunk, olyan formátumokban, amelyek már messze túlmutatnak a „hagyományos" internetes műsorszórás eddig ismertetett alapjain (*lásd a 2. ábrát*). Az elterjedt MPEG-1, MPEG-2, MPEG-4 vagy éppen DVD formátumú mozgóképadás mellett ugyanis digitális műholdas adásra vagy földi sugárzású digitális csatornák jelének alapjául szolgáló adatfolyamok szórására is lehetőség nyílik. Ezekhez természetesen különleges átjárókra van szükség, hogy mindez egy PC-ről lehetővé váljon. A DVB (Digital Video Broadcasting) formátumnak több változata is létezik. A DVB-S szabvány a műholdas műsorszórást, a DVB-C a kábeltelevíziós műsorszórást, míg a DVB-T a földi digitális műsorszórást jelenti, emellett akár LAN hálózatokon is továbbíthatjuk DVB formátumú mozgóképeinket. A műsorfolyamot tekintve mindegyik szinte ugyanolyan, kizárólag a továbbítás módja az, amelyben az egyes szabványok eltérnek egymástól. A műsorszórás természetesen a rádióadóknál ismertetett sémára épül, mondhatni teljesen ugyanaz. Maga a program két részből áll: az egyik a kiszolgálórész, másik a lejátszórész – érdekessége, hogy a lejátszó is használható kiszolgálóként.
A csomag egyik érdekessége, hogy képes csatornaadat-szolgáltatás kezelésére. Ez annyit jelent, hogy az adásokhoz hozzáadható némi szöveges adat, például a csatorna neve, profilja vagy bármi, ami a műsorral kapcsolatos. Minderről egy mini-SAP nevű program gondolkodik, amely a SAP/SDP szabvány használatával küldi el a csatorna adatait a vevőhöz. A videoLAN kiszolgáló a hagyományos egyéni (unicast) módú üzenetküldés mellett a csoportos (multicast) módú üzenetküldést is támogatja, emellett teljes IPv6-támogatást biztosít, amely a jövőre és a csoportos üzenetküldés fontosságára nézve igen előnyös.
A programcsomag mindezek felett még egy különleges műsorszórási eljárást is támogat, amely bár nem illik bele az eddig ismeretett valós idejű műsorszolgáltatások sorába, de valószínűleg minden felhasználó álma. Ez nem más, mint a Video On Demand (betűszóval VOD), ami a nevéből is kiolvashatóan igény szerinti videózást jelent. Ez a gyakorlatban annyit tesz, hogy a felhasználó vagy a vevő az adott műsort (például egy filmet) bármikor elindíthatja, nézés közben megállíthatja, előretekerhet, visszatekerhet. A szolgáltatás egyetlen hátránya az igen magas erőforrásigény, hiszen ezt a fajta adatfolyamot csak egyéni módú üzenetküldéssel lehet továbbítani. Arról nem is beszélve, hogy a kiszolgáló is megterhelődik, hiszen egyszerre akár több száz szálon kell merevlemezéről olvasni az adatokat, nem is szólva az egyéb terhelést jelentő kérésekről. A leírás szerint a megoldás lényege, hogy HTTP protokollon keresztül történik az adatszolgáltatás, ehhez csupán egy webkiszolgálóra van szükség, amely az adatfolyamot HTTP-csomagok formájában továbbítja, a HTTP 1.1 protokoll pedig a pozícionálást teszi lehetővé az adatfolyamban (így működnek a letöltést gyorsító programok is), ezt használják fel az előre- és visszapörgetésre. A lejátszáshoz csupán a csomag VLC nevű programjára van szükség, amely igen egyszerűen elindítja az általunk kért műsort.
Mint láthatjuk, az internetes műsorszórás feltételeit megteremtő programok palettája meglehetősen színes, bevallom, magam is meglepődtem, amikor a témába beleásva magamat a vártnál jóval több megoldást találtam, többet, mint amelyek más operációs rendszerek esetében rendelkezésre állnak. Hogy ezt az előnyt sikerüljön megőrizni, a cikksorozat következő részeiben a fent említett programokat szeretném a lehető legnagyobb részletességgel bemutatni, így nem csak tippeket fogunk kapni a témával kapcsolatban, hanem minden eszközünk meglesz ahhoz, hogy a rendelkezésünkre álló programokat hatékonyan is tudjuk használni.

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

Írjunk többszálú programot Linux alatt! (2. rész)



Sorozatunkat a QThread és a többi szálkezelő osztály áttekintésével folytatjuk.

Az elmúlt hónapban a többszálú programok készítésének és a Qt környezet által nyújtott lehetőségek alapjaival foglalkoztunk. Most folytatjuk a Qt többszálú programok készítését támogató moduljának bemutatását, célunk a részletek pontosabb megismertetése.
Minden szálosztálynak a QThread az őse, ezért nézzük meg egy kicsit részletesebben is!

Public eljárások

- `bool wait(unsigned long time = ULONG_MAX)` – a szálak összekapcsolását (JOIN) teszi lehetővé.
  Az előző részben (Linuxvilág, októberi szám, 51. oldal) már részletesebben ismertettük a működését. Amikor a `time` valamilyen tényleges ezredmásodperc érték, akkor ez az eljárás csak a megadott ideig várakozik a meghívott szálra.
- A `wait()` sikertelen összeláncolás esetén `false` értéket ad vissza.
- A `void start()` – elindítja a szál párhuzamosan futó kódját (lásd `run()` tagfüggvény).
- A `bool running() const` – a szál fut, illetve nem fut állapotát kérdezhetjük le.

Statikus public eljárások

- `void postEvent (QObject *receiver, QEvent *event)`
  A Qt legfontosabb része természetesen a grafikus GUI-készítési lehetőség. A jel/foglalat (signal/slot) szerkezet nagyon érdekes tervezési minta, de a Qt lehetőséget ad az aszinkron eseménykezelésre is. Ez az eljárás egy fogadó grafikus elem (widget) számára tud egy eseményt szálbiztosan elküldeni.
- `void exit()` – leállítja az ezt a tagfüggvényt hívó szál futását, valamint felébreszti az összes olyan szálat, ami eddig ennek a szálnak a lefutására várt.

Protected eljárások

- `virtual void run()`
  A Java nyelvhez hasonlóan itt is a `run()` tagfüggvény képviseli a szál kódját, amit nekünk kell mindig újra megvalósítani.

Statikus protected eljárások

- `void sleep(unsigned long masodperc)`
- `void msleep( unsigned long ezred_masodperc)`
- `void usleep( unsigned long millomod_masodperc)`

Ezek az eljárások a megadott ideig felfüggesztik a szálak futását (a szálakat elaltatják), ez egyben a hívó szál futási jogáról való lemondását is jelenti. Egy nagyon rövid várakoztatáshívás az együttműködő szálfuttatási rendszerek klasszikus `yield()` (azaz lemondok a futási jogomról) függvényének felel meg.

A közösen használható erőforrások védelme

Az eddigi két program nem használt minden szál által látható, közös (globális) erőforrásokat, hiszen csak helyi változóink voltak. A párhuzamos programozás megvalósításához a szál fogalma nem elegendő, hiszen biztosítani kell a közös erőforrások megfelelő sorrendben (összehangolás) való használatát. Gondoljunk a közúti közlekedés példájára! Képzeljük el, hogy az egyes szálak a közlekedési eszközök vagy a gyalogosok. Ekkor a közösen használt erőforrások az útkereszteződések és a zebra. A KRESZ szabályai és a jelzőlámpák valósítják meg ebben a rendszerben az összehangolást. A Qt egyik összehangoló eszközét a `QMutex` osztály képviseli.
Az osztály célja a közösen használható objektumok védelme, illetve a forrásprogramok kényes szakaszainak kijelölése.
A `QMutex` kétállapotú sorompó (nyitott vagy zárt) módjára véd. Amennyiben egy `mutex` zárt, úgy az általa védett kódrészbe a többi szál nem léphet be, azaz várakozniuk kell.
Lássuk a `QMutex` osztályt!

- `QMutex(bool recursive = FALSE)`
  Létrehoz (constructor) egy `mutex` objektumot. Lehetőség nyílik a `recursive` beállítás használatára, ugyanakkor a `TRUE` érték megadása szükséges, amely esetben a `mutex`-re kiadott `lock()` hívások száma nyilvántartódik, és pontosan ugyanennyi `unlock()` meghívására lesz szükség a `mutex` felnyitásához.
- `void lock()` – lezárja a `mutex` objektumot. Amennyiben a `mutex` objektumot egy másik szál már lezárta, úgy ez a tagfüggvény blokkolja a hívó szálat, ami csak akkor futhat tovább, ha a `mutex` ismét nyitott állapotba kerül és utána sikerül a zárolás.
- `void unlock()` – felnyitja a `mutex` objektumot.
- `bool locked()` – visszaadja, hogy egy `mutex` objektum most éppen zárt vagy nyitott állapotú-e.
- `bool tryLock()` – a célja hasonló a `lock()` tagfüggvényéhez, de sikertelenség esetén a hívó szál nem blokkolódik, ezért programunknak lehetősége van egy másik logikai ágon továbbfutnia.

A Qt csomag egy másik összehangolási fogalmat is ad: a `QSemaphore` osztályt. Egy `QMutex` objektum kétállapotú jelzőnek (semaphor) feleltethető meg. Általános értelemben viszont jól jön egy olyan eszköz is, ami nemcsak egyetlen szálat enged be a védett területre, hanem annyit, amennyit a jelző létrehozójában (constructor) megadtunk. Miért jó ez nekünk? Képzeljük el, hogy egy vendéglőben *N* darab szék van, és a vacsoravendégek jelképezik az egyes szálakat. Ekkor úgy kell szervezni a vendéglő működését, hogy egyszerre legfeljebb csak *N* darab vendég mehessen be az ajtón, a többi kint várakozzon. Ha néhány vendég elhagyja a vendéglőt, akkor a várakozók közül már kiválaszthatók a következő szerencsés és éhes vendégek. A számítástechnikához kicsit közelebb álló példa lehet a TCP/IP-kapcsolatkészlet esete. Tegyük fel, hogy van 10 előre elkészített `Connection` objektumunk.

Ekkor egy jelző vezérelheti a szálak kapcsolatkérési igényeit. A `QSemaphore` osztály rendelkezik az atomi módon végrehajtható növelő, csökkentő műveletekkel, amiket részletesen a Qt leírásából tanulmányozhatunk.

Az író–olvasó feladat megoldása

Gyakori helyzet, hogy létezik egy osztottan használható erőforrásunk (egy adatbázis, egy memóriaváltozó), amit néhány írófolyamat időnként frissít (általában egy lassú író művelettel), illetve sok olvasófolyamat olvas (ez általában egyetlen gyors művelet). Az nyilvánvaló, hogy az erőforrás írása során nincs értelme olvasni, hiszen ekkor átmeneti állapotban lehet a kiolvasandó adat. Ugyanakkor több olvasószál természetesen párhuzamosan olvashat. Ilyen jellegű gond merülhet fel például egy internetes tőzsdei index szolgáltató alkalmazás esetén. Példaképpen legyen a feladatunk az, hogy egy integer változót (neve: `TheValue`) írunk és olvasunk párhuzamosan. Egy lehetséges megoldást mutat be az *1. listán* található forrásprogram.
A 15–20. sorokban létrehozott `kiir()` függvény biztosítja azt, hogy a képernyőre írás megszakíthatatlan (atomi) művelet legyen. Ezt a kényes szakaszt a `kiiroMutex` védi. A Java-programozók ilyen kóddal oldották volna meg a feladatot:

```
synchronized ( kiirastVedo
_ Objektum ) {
kiiras...
}
```

A `countOfReader` változó tárolja a pillanatnyi olvasók számát. A 31. sorban látható létrehozónak egy egész értékű szálazonosítót tudunk átadni. A `TReaderWriter` osztály két lényegi tagfüggvénye a `reader()` és a `writer()`. Ezek működését a `reader_mutex` és a `writer_mutex` objektum hangolja össze. A `writer()` tagfüggvényt teljes mértékben védi a `writer_mutex`. A `TheValue` értékét azért változtattuk meg öt lépésben, hogy életszerűbb legyen, ugyanis az írási folyamat általában több lépésből áll, és emiatt a kényes szakasz bevezetése nélkül könnyen megszakítható lenne. Ekkor pedig a `TheValue` változó öttel nem osztható állapota jöhetne létre. A 44. sorban kezdődő `reader()` működése azon az ötleten alapul, hogy az olvasó szálakat számoljuk, és amennyiben a számuk nulla, úgy a `writer_mutex`-et is lezárt állapotba hozzuk. Lehetséges, hogy a `writer_mutex.lock()` nem sikerül, de ez pont jó, hiszen ekkor egy írófolyamat dolgozik és nincs is értelme addig olvasni. Ekkor itt várunk, amíg az író be nem fejezi a munkáját. Az is lehetséges, hogy a `writer_mutex.lock()` sikerült. Ezután természetesen az írók fognak várni addig, amíg az olvasók száma nulla nem lesz, mert a 60–63. sorok tanúsága szerint csak ekkor történik meg ismét a `writer_mutex` felnyitása.
A 84–101. sorokban megvalósított `run()` tagfüggvény az eddigiek alapján már biztosan érthető, a feladata az, hogy egy szál `reader()` és `writer()` műveletet hajtson végre.
A `main()` függvényben egy tízelemű, `TReaderWriter` mutatókból (pointer) álló tömböt hozunk létre, majd elindítjuk őket egy-egy új `TReaderWriter` objektummal, majd elindítjuk a tíz végrehajtási szálat. A 121. sorban egy tetszőleges billentyűleütésre várunk, ennek hatására szabályosan befejeződik a programunk.

A szálak elaltatása és felébresztése: feltételes várakoztatás

A párhuzamosan futó szálak különféle feladatokat hajthatnak végre, de általában kijelenthető az is, hogy a teendőknek valamilyen előfeltételük szokott lenni. Amikor egy szál futási feltétele nem teljesül, elaltathatja magát. Egy másik szál figyeli azt, hogy a feltétel most már teljesül-e, ha igen, az alvó szálat felébreszti. Érezhető, hogy ebben a működésben a `QMutex` által kialakított szál összehangolásának is jelentős szerep jut, hiszen arra is vigyázni kell, hogy a felébredt szál futási feltételei ne váljanak ismét hamissá. A Qt a `QWaitCondition` osztály segítségével teszi elérhetővé a feltételes várakoztatást, ezért nézzük meg, mit nyújt számunkra ez az osztály!

```
bool wait(unsigned long time = ULONG_MAX)
```

Elaltatja a szálat, ami az alapértelmezett `ULONG_MAX` érték esetén addig alszik, amíg valaki fel nem ébreszti. A `wait()` tagfüggvényben valós időt is megadhatunk (ezredmásodpercben), amely time-outként működik, azaz ha a megadott idő letelik, a szál magától felébred.

```
bool wait(QMutex *mutex, unsigned long
➥time = ULONG_MAX)
```

Hasonlóan működik, mint az előző `wait()` tagfüggvény, de elalvás előtt kinyitja a zárolt `mutex`-et.

```
void wakeOne()
void wakeAll()
```

Legyen `qwc` egy `QWaitCondition` típusú változó (objektum). Ekkor a `qwc.wakeOne()` felébreszt egy olyan szálat, amit előzőleg egy `qwc.wait()` hívás altatott el. A `qwc.wakeAll()` az összes ilyen szálat felébreszti. A feltételes elalvás–felébredés jobb megértése érdekében nézzünk meg egy olyan példát, ahol három dolgozó szál csinál valami hasznosat (példánkban csak egy üzenetet jelenít meg a képernyőn, ezt most nem hangoljuk össze egy `kiir()` tagfüggvénnyel). A `main()` függvény szála (azaz a fő szál) egy gombnyomáseseményt figyel, ennek bekövetkeztekor felébreszti az erre a feltételre várakozó három dolgozószálat (2. lista, 53. CD Magazin/Qt könyvtára).
A 12. sorban meghatározott gombnyomásváltozóval kezeljük a feltételes elalvást, illetve felébredést. Itt jó programozási szokás, ha az ilyen változónak a feltétel vagy az esemény nevét adjuk meg. A 43. sorban látható, hogy a `TDolgozo::run()` szálkód végrehajtása lényegében azzal kezdődik, hogy egy gombnyomás eseményre várva elaltatja magát. A `main()` szál ciklusfeltételében egy gombnyomásra várakozás van.
A `while` ciklus mindaddig ismétlődni fog, amíg nem a 'v'-t (v=vége) nyomtuk le. Egy nem v billentyű lenyomása után a főprogramban (82. sor) a `gombnyomas.wakeAll()` felébreszti mindhárom dolgozószálat, hiszen mindegyik a `gombnyomas.wait()` hívásnál aludt el. A dolgozó szálak (d1, d2, d3) az idő haladtával ismét el fognak aludni, ha már mindegyik alszik, akkor az `ennyiSzalDolgozik` változó nulla lesz, így a fő szál belső `while` ciklusából újra és újra kiléphetünk, és az összes dolgozót felébreszthetjük.

A termelő-, illetve fogyasztófeladat megoldása

A párhuzamos programozás másik klasszikus kérdése (nevezhetnénk desing patternnek, azaz tervezési mintának is) a termelő-, illetve fogyasztófeladat. Van egy raktár, ahová a termelőszál bizonyos típusú objektumokat helyez el, míg ezeket a fogyasztószál kiveszi innen, hogy valahol majd felhasználja őket. A raktár véges méretű, azaz legfeljebb ***K*** darab objektum befogadására alkalmas. Az összehangolásnak két feltétele van:

```
1. //
2. // Író/olvasó feladat
3. //
4.
5. #include <stdio.h>
6. #include <stdlib.h>
7. #include <iostream>
8. #include <string>
9. #include <qthread.h>
10.
11.using std::cout;
12.
13.static QMutex kiiroMutex;
14.
15.void kiir(char *sz, int th_szam,
            ➥int ertek)
16.{
17. kiiroMutex.lock();
18. cout << "\n" << sz << " (szál=" <<
    ➥th_szam << ", érték=" << ertek << ")";
19. kiiroMutex.unlock();
20.}
21.
22.class TReaderWriter : public QThread
23.{
24. static QMutex reader_mutex;
25. static QMutex writer_mutex;
26. static int countOfReader; // Az
olvasószálak pillanatnyi száma
27. static int TheValue; // Az olvasandó
    ➥érték tárolója
28. int th_ID; // A szál azonosítója
29. bool leallitott;
30.public:
31. TReaderWriter( int pID )
    ➥{ th_ID = pID; }
32. void reader();
33. void writer();
34. void leallit();
35. virtual void run();
36.};
37.
38.
39.QMutex TReaderWriter::reader_mutex;
40.QMutex TReaderWriter::writer_mutex;
41.int TReaderWriter::countOfReader = 0;
42.int TReaderWriter::TheValue = 0;
43.
44.void TReaderWriter::reader()
45.{
46. reader_mutex.lock();
47. if ( countOfReader == 0 )
48. {
49. writer_mutex.lock();
50. }
51. countOfReader++;
52. reader_mutex.unlock();
53.
54. kiir("Egy ovasás történt:", th_ID,
        ➥TheValue);
55.
56. reader_mutex.lock();
57. countOfReader--;
58. reader_mutex.unlock();
59.
60. if ( countOfReader == 0 )
61. {
62. writer_mutex.unlock();
63. }
64.}
65.
66.void TReaderWriter::writer()
67.{
68. writer_mutex.lock();
69. TheValue++;
70. TheValue++;
71. TheValue++;
72. TheValue++;
73. TheValue++;
74. kiir("Az er _ forrás értéke növelve:",
        ➥-1, TheValue);
75. writer_mutex.unlock();
76.}
77.
78.void TReaderWriter::leallit(void)
79.{
80. leallitott = true;
81.}
82.
83.
84.void TReaderWriter::run()
85.{
86. leallitott = false;
```

folytatás a következő oldalon

- A termelőnek aludnia kell addig, amíg a raktár tele van.
- A fogyasztónak aludnia kell addig, amíg a raktár teljesen üres.

Példaprogramunkban a raktárat egy `BUFFER_MERET` nagyságú, egészekből álló tömb testesíti meg. A termelt objektumok egész számok. A program ismertetése előtt kiemelném két érdekességét. Az első az, hogy a raktár egy ciklikus FIFO segítségével van megvalósítva. Ebben az adatszerkezetben a betett elemek számát is figyelemmel kell kísérnünk, különben az `ÜRES` és a `MEGTELT` eseteket nem tudnánk megkülönböztetni. A másik érdekesség, hogy a program az összehangolt adatkezeléshez (raktárkezeléshez) a monitorelgondolást használja. Ez a tervezési minta egységbe zárja a védett erőforrást (adatot) és az ezt kezelő műveleteket (függvényeket). A szálak az erőforrást csak a monitor által biztosított műveleteken keresztül kezelhetik. (3. lista, 53. CD Magazin/Qt könyvtára.)

A forráskód elején lévő `BUFFER_MERET` állandó (constans) a raktár nagyságát, a `TERMELENDO` pedig a legyártandó egész számok számát határozza meg. A program 24–29. sorában megvalósítottuk az összehangolt képernyőre írást, a szálak csak ennek a használatával jeleníthetik meg a szöveges adatokat. A 30. sortól egy segédfüggvényt találunk, ami a *libc* véletlenszám-generáló lehetőségét használja egy új (1 és 5000 közötti) egész érték létrehozásához.

folytatás az előző oldalról

```
87.
88. reader_mutex.lock();
89. kiir("Elindult egy új szál:", th_ID, -1);
90. reader_mutex.unlock();
91.
92. while ( !leallitott )
93. {
94. for(int i = 0; i < 20; i++)
95. {
96. reader();
97. }
98. writer();
99. }
100. kiir("Leált a következ _ szál:",
          ➥th_ID, -1);
101.}
102.
103.
104.//
105.// Indulás...
106.//
107.int main()
108.{
109. TReaderWriter *rw[10];
110.
111. for (int i=0; i<10; i++)
112. {
113. rw[i] = new TReaderWriter(i);
114. }
115.
116. for (int i=0; i<10; i++)
117. {
118. rw[i]->start();
119. }
120.
121. getchar();
122.
123. for (int i=0; i<10; i++)
124. {
125. rw[i]->leallit();
126. }
127.
128. for (int i=0; i<10; i++)
129. {
130. rw[i]->wait();
131. }
132.
133. for (int i=0; i<10; i++)
134. {
135. delete rw[i];
136. }
137.
138. return 0;
139.}
```

A 39. sortól található meg a `TMonitor` osztály felülete és megvalósítása.

```
39. class TMonitor
40. {
41. int buffer[BUFFER_MERET]; // Ide termelünk
    és innen fogyasztunk
42. int elso, utolso, betettElemekSzama;
43. public:
44. TMonitor();
45. void betesz(int e);
46. int kivesz(void);
47. };
```

Ez a típus lesz az a monitor, amelyik átmenetitárazza a termelő termelését, illetve szolgáltatja az objektumokat (most csak egy-egy egész szám) a fogyasztónak. Ezt a két műveletet a `betesz()` és a `kivesz()` tagfüggvények valósítják meg. Vegyük észre, hogy a `betesz()` a fogyasztószálat, a `kivesz()` a termelőszálat ébreszti, bár lehet, hogy erre nem lenne szükség, de nem lesz belőle baj. Az az igazság, hogy az így, feltétel nélkül kiadott `wakeOne()`, azaz felébresztés valószínűleg nem nagyobb költségű, mintha megvizsgálnánk, hogy a másik szál alszik-e. A 98–102. sorig a termelőosztály, míg a 107–111. sorig a fogyasztóosztály felülete van meghatározva. Látható, hogy mindkettőben csak a `run()` tagfüggvény került újraírásra. Mindkét `run()` tagfüggvény annyira egyszerű, hogy nem is érdemes hozzájuk magyarázó szöveget írni. A fő szálat megvalósító `main()` függvény pedig még ennél is egyszerűbb. Röviden ennyit szerettem volna elmesélni a Qt többszálúság (multithread) lehetőségeiről. A példaprogramok tanulmányozásához sok sikert kívánok.

Az összes példa egyszerre történő lefordításához mellékeltem egy `Makefile`-t. Nézze meg mindenki, hogy nála is a */usr/lib/qt3* könyvtár tartalmazza-e a Qt csomagot! Ha nem, akkor a `Makefile`-ban a jó elérési útra kell beállítani a `-I` és `-L` kapcsolókat. A `make` parancs kapcsolók nélküli futtatásával az összes példa futtatható változata előállítható.
Mindenkinek kellemes programozást kívánok!

A cikkhez tartozó listák megtalálhatóak az 53. CD Magazin/Qt könyvtárában.

Nyíri Imre (inyiri@mol.hu)
Jelenleg a MOL Rt.-nél dolgozik. Informatikai vállalkozásában az Internet, a Linux, valamint a Java-programozás gyakorlati hasznosításával foglalkozik. Örök szerelme a C++ maradt.

KAPCSOLÓDÓ CÍMEK

1. A Qt-csomaghoz adott leírás
   ➲ http://doc.trolltech.com
   ➲ http://www.trolltech.com
2. Dr. Kacsuk Péter–Ferenczi Szabolcs
   Párhuzamos és konkurens programozás soktranszputeres rendszereken
   Budapest, 1993 BME jegyzet (ISBN 963 431 774 X)

Héjprogramozás Linux alatt (7. rész)

Akármilyen nyelven fejlesztünk is programot, egy idő után számos olyan építőelemet fedezünk fel a saját munkánkban, amit újra és újra felhasználunk.

Mostanra e sorozatunkban is eljutottunk odáig, hogy elkezdhetjük áttekinteni a héjprogramozás kifinomultabb módszereit, valamint a programban gyakran felbukkanó, „újrahasznosítható” kódelemeket.

A parancssori kapcsolók kezelése

A sorozat egy korábbi részében már megtanultuk, hogyan lehet a parancssori kapcsolókat kezelni. Tudjuk tehát, hogy ezekhez a `$1`, `$2` stb. szimbólumok segítségével férhetünk hozzá. Mármost a parancssori kapcsolók is alapvetően két csoportra oszthatók: vannak az „igazi” kapcsolók, amelyek a program által megvalósított feldolgozási művelet célját vagy forrását határozzák meg, és léteznek az értékek, amelyek a feldolgozás miként jét szabályozzák. Tekintve, hogy Unix alatt minden fájlnak minősül, az értékek általában fájlnevek, a kapcsolók pedig egybetűs parancsok, amelyeket a Unix hagyományainak megfelelően általában egy – (mínuszjel) előz meg.

Ahhoz, hogy programunk a kapcsolók hatására a megfelelő műveletsort hajthassa végre, rendelkeznie kell egy olyan résszel, amely a kapcsolókat felismeri és a végrehajtást a megfelelő mederbe tereli.

Tegyük fel, hogy programunk működése legfeljebb három különböző kapcsolóval vezérelhető. Ezek a `-a`, `-b` és `-c` szimbólumok legyenek. Ezenkívül a kapcsolók, illetve a fájlnevek megadásának sorrendjére vonatkozóan semmiféle további megkötéssel nem élünk: például a kapcsolókat egybe szabad írni, vagyis a `-a -b` formátum helyett azonos jelentéssel használható a `-ab` is. Továbbá egy vagy több fájlnevet is meg lehet adni kapcsoló előtt és után, vagy akár két kapcsoló között is, sőt egy kapcsoló akár többször is szerepeltethető. Összefoglalva tehát, a felhasználó gyakorlatilag azt tesz, amit akar, a programnak kell elég intelligensnek lennie ahhoz, hogy bármilyen formátumú parancssort helyesen értelmezzen. (Egyetlen dolgot azért mégis kössünk ki: fájl neve soha ne kezdődjön mínuszjellel.)

Milyen elemekből kell tehát állnia egy ilyen programnak? Először is a megoldandó feladat két, világosan elkülönülő részre választható szét: azonosítanunk kell a kapcsolókat, és össze kell gyűjtenünk a fájlneveket. Ha két művelet ennyire élesen elkülönül, célszerű két függvény formájában megvalósítani őket.

A parancssorba a felhasználó által „beömlesztett” értékek szétválogatásának alapelve nyilván az lehet, hogy a kapcsolók vagy azok csoportjai mínuszjellel kezdődnek, a fájlnevek pedig nem. Ha fájlnevet találunk valahol, azt elegendő egy nagy listába begyűjteni, ami viszonylag könnyen megoldható, ha tudjuk, hogy héjprogramokban a karakterláncok összemásolásához nem kell egyebet tennünk, mint egy értékadás jobb oldalán egymás mellé írni őket.

A kapcsolóknál eleve fel kell készülünk rá, hogy a felhasználó esetleg csoportokban adta meg őket, így a legegyszerűbb megoldás az, ha egy-egy `grep` parancs segítségével minden mínuszjellel kezdődő parancssori kapcsolóban az összes lehetséges kapcsolót végigvizsgáljuk, illetve az esetlegesen jelenlevő érvénytelen kapcsolókat is. Ennyi bevezető után lássuk a feladat egyik lehetséges megoldását!

```
 1: #!/bin/sh
 2: PROGRAMNEV=`basename $0`
 3:
 4: # Kapcsolók kigyűjtése
 5: kapcsolok()
 6: {
 7:   for i in $*
 8:   do
 9:     if echo $i | grep ^- > /dev/null
10:     then
11:     # Hibás kapcsoló keresése
12:       if echo $i | grep '[^-abc]'
                      ➥> /dev/null
13:       then
14:         return 1
15:       fi
16:     # Érvényes kapcsolók keresése
17:       for kapcsolo in a b c
18:       do
19:         if [ `echo $i | grep $kapcsolo |
                  ➥wc -l` -eq 1 ]
20:         then
21:           parancs="$kapcsolo=1"
22:           eval "$parancs"
23:         fi
24:       done
25:     fi
26:   done
27: }
28:
29: # Fájlnevek kigyűjtése
30: fajlnevek()
31: {
32:   lista=""
33:   for i in $*
34:   do
35:     if echo $i | grep -v ^- > /dev/null
36:     then
37:       lista=$lista" "$i
38:     fi
39:   done
40:   echo $lista
41: }
42:
43: # Tájékoztató szöveg
44: if [ $# -eq 0 ]
45: then
46:   echo "Használat: $PROGRAMNEV [-abc]
            ➥<fájlnév>"
47:   exit 1
48: fi
49:
50: # Kapcsolók vizsgálata
```

```
51: a=0; b=0; c=0
52: kapcsolok $*
53: if [ $? -ne 0 ]
54: then
55:   echo "Hibás kapcsoló!" ;  exit 1
56: fi
57:
58: # Fájlnevek kigyűjtése
59: fajlok=`fajlnevek $*`
60:
61: echo "Kapcsolók beállításai: a=$a b=$b
        ➥c=$c"
62: echo "A feldolgozandó fájlok listája:
        ➥$fajlok"
```

A kapcsolókat a `kapcsolok()`, a fájlok neveit pedig a `fajlnevek()` függvény kezeli. Bár a főprogramban ezt hívjuk meg később, kezdjük a vizsgálatot az utóbbival. Mindenekelőtt figyeljük meg, hogy a függvény meghívásakor (59. sor) a program teljes paraméterkészletét (`$*`) átadjuk neki. Erre azért van szükség, mert – mint arról korábban már volt szó – a függvény „program a programban”, vagyis mindenből sajátja van. Létezik saját bemenete és kimenete, valamint saját parancssora is. Ha ide nem emeljük át az egész héjprogramnak átadott értékeket, akkor a függvény nem fogja látni őket. Számára ugyanis `$1` nem a főprogram első parancssori értékét jelenti, hanem a sajátját.
A végrehajtandó műveletsor lelke a 35. sorban látható. Itt megvizsgáljuk, hogy a 33. sorban induló `for` ciklus által pillanatnyilag kiválasztott érték első karaktere mínuszjel-e vagy sem. Ha nem (a negálást a `grep -v` kapcsolója végzi), akkor a tartalmát egy szóközzel kiegészítve a `lista` nevű gyűjtőváltozóhoz másoljuk. A ciklus lefutása után ennek a tartalmát adjuk vissza egy közönséges `echo`-val a függvény szabványos kimenetén keresztül. Megint fontos hangsúlyozni, hogy a függvény csatornájáról van szó, vagyis a fájlok listája nem a képernyőre kerül. Éppen ennek a kimenetnek az elfogására, és a `fajlok` nevű változóban való elhelyezésére szolgál az 59. sorban a parancsbehelyettesítés.
A kapcsolók vizsgálatánál hasonló módszereket alkalmazunk. Ha egy kapcsoló meg van adva a parancssorban, akkor egy, a nevével azonos héjváltozó értékét 1-re állítjuk. (Természetesen bármilyen más módszer is használható lenne.) A `kapcsolok()` függvénynek átadott értékeken a 7. sorban induló `for` ciklus megy végig. Minden mínuszjellel (-) kezdődő csoportnál először célszerűen azt kell megvizsgálnunk, hogy tartalmaz-e téves kapcsolót (`grep '[^-abc]'`). Figyeljük meg, hogy itt és valamennyi ehhez hasonló vizsgálatnál a `grep` kimenetét a */dev/null*-ba irányítottuk. Erre azért van szükség, mert a `grep` alapértelmezett viselkedése szerint a találatokat kiküldi a szabványos kimenetére. Ez most azonos lenne a függvény szabványos kimenetével, és mivel az sincs átirányítva, a szöveg végül a képernyőre kerülne. Ha hibás kapcsolót találtunk, azt a függvény azonnali visszatérésével és az 1-es visszatérési értékkel (`return 1`) jelezzük. Ez utóbbit az 53. sorban a `$?` változó vizsgálatával detektálja a főprogram. (A `$?` beépített héjváltozó a legutoljára lefutott parancs visszatérési értékét tartalmazza. Jelen esetben a `kapcsolok()` függvény egyetlen parancsnak minősül.)
A 16–23. sorban immár az érvényes kapcsolókat vizsgáljuk. Egy `for` ciklussal egyenként végigpróbáljuk valamennyit, és ha illeszkedés van az adott csoporton belül, akkor a megfelelő nevű változó értékét `1`-re állítjuk. Ezt az utóbbi műveletet egy kicsit trükkösen kell végrehajtanunk, hiszen most a beállítandó változó nevét maga a ciklusváltozó (`$kapcsolo`) tartalmazza. Többek között az ilyen közvetett értékadások megoldására szolgál az `eval` parancs, amely egy karakterláncban megadott parancssort (jelen esetben a `$parancs`-ot) hajtja végre.

A szabványos megoldás: getopts

A parancssori kapcsolók kezelése annyira általános feladat, hogy maga a Unix, illetve a héj is rendelkezik egy külön erre a célra szolgáló `getopts` nevű paranccsal, ennek használata a következő:

```
getopts "lehetséges_kapcsolók" változónév
```

A `getopts` tulajdonképpen pontosan ugyanazt teszi, mint az előző programunk, vagyis a megadott mintában előforduló betűket keresi a mínuszjellel kezdődő parancssori értékekben. Ha érvényes kapcsolót talál, azt elhelyezi a megadott héjváltozóban. Ilyenkor a visszatérési értéke igaz, egyébként hamis. Ezt az utóbbi tulajdonságát használhatjuk ki arra, hogy egy `while` ciklussal az összes kapcsolót végignézessük vele. (A `getopts` mindig csak a soron következő kapcsolót vizsgálja, de megjegyzi, hogy hol tart.)
Ha hibás kapcsolót talál, akkor alapértelmezés szerint hibaüzenetet küld a képernyőre, a munkaváltozóba pedig egy ?-t (kérdőjelet) tesz. A hibaüzenet kiküldését megtilthatjuk, ha a lehetséges kapcsolók felsorolását egy kettősponttal kezdjük. Mindezek ismeretében előző programunkat a következőképpen is megírhatjuk:

```
 1: #!/bin/sh
 2: # A getopts parancs használata
 3:
 4: PROGRAMNEV=`basename $0`
 5:
 6: if [ $# -eq 0 ]
 7: then
 8:   echo "Használat: $PROGRAMNEV [-abc]
          ➥<fájlnév>"
 9:   exit 1
10: fi
11:
12: while getopts ":abc" KAPCSOLO
13: do
14:    case $KAPCSOLO in
15:      "a") echo "-a kapcsoló megadva";;
16:      "b") echo "-b kapcsoló megadva";;
17:      "c") echo "-c kapcsoló megadva";;
18:      "?") echo "Hibás kapcsoló"; exit 2;;
19:    esac
20: done
```

A rövidség ára a kötöttebb működés. A kapcsolókat továbbra is egybe szabad írni, de a `getopts` azt elvárja tőlünk, hogy a fájlok felsorolását valamennyi kapcsoló megadása megelőzze. Előző, lényegesen hosszabb programunk ugyanakkor e tekintetben is teljesen „liberális” volt.

Büki András (buki.andras@insilico.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.

Vírusok, férgek, hátsó ajtók, trójaiak (16. rész)

Az előző részben az operációs rendszerek védelmi módszereiről esett szó, azaz megnéztük, miként tárolja a rendszer, hogy ki milyen erőforrásokhoz férhet hozzá, és azokkal mit tehet. A biztonság kérdése azonban itt nem merül ki. Hiába jó egy védelmi módszer, a rendszerre számtalan más veszély is leselkedik.

Ez a rész egy kicsit rendhagyó lesz, mivel olyan dolgokról ejtünk szót, amelyek nem feltétlenül esnek bele az operációs rendszert alkotó „fogaskerekek” ismertetésébe. Erre mégis szükség van, mert a rendszer biztonságát olyan dolgok is fenyegethetik, amelyek ellen az operációs rendszer egymaga nem tud védekezni – a felhasználónak is tennie kell az ügy érdekében. Ez azonban visszafelé is igaz: hiába tesz meg mindent a felhasználó, ha az operációs rendszer nem biztonságos, nem ér semmit az egész. A biztonság pedig nem egy utólag is hozzáadható tulajdonság. Erre már akkor készülni kell, amikor lerakjuk a rendszer első „alapköveit”.

Vegyük például a vírusokat: az operációs rendszer számára a vírusok ugyanolyan programok, mint a szövegszerkesztő vagy a parancsértelmező. A víruskereső alkalmazás is felhasználói szinten fut. Mégis létezik olyan operációs rendszer, amelyik kevésbé van kitéve a vírusok támadásának. A Unix-rendszerekben például a felhasználók nem módosíthatják a programok kódját, így a vírus nem képes további alkalmazásokat megfertőzni. Az MS-DOS-felhasználók azonban hiába rettegnek a vírusoktól és tennének meg mindent a fertőzés elkerülése végett, ilyenfajta védelemről nem is álmodhatnak. Hiába használunk azonban Linuxot, ha rendszergazdaként indítgatunk mindenféle kétes származású alkalmazást – a fertőzés veszélye emiatt ugyanúgy fennáll.

Még mielőtt azonban elmélyednénk a férgek és a vírusok természetében, az előző részből maradt adósságunkat törlesztjük.

Felhasználók hitelesítése

Mi, emberek olyan okosak vagyunk, hogy embertársainkat könnyedén felismerjük. A gépek azonban buták, ma még nem képesek úgy felismerni a felhasználót, mint ahogy mi a szomszédunkat a liftben. Ezért az operációs rendszernek nincs más választása: meg kell kérdeznie, hogy éppen ki akarja őt használni. Egyes rendszerek feltétel nélkül elhiszik, amit a felhasználó mond, más rendszerek azért ellenőriznék valamilyen módon. Ezt az eljárást nevezzük hitelesítésnek.

A hitelesítés három dolgon alapulhat: vagy a felhasználó tudását ellenőrizzük (jelszó), vagy a tulajdonát (mágneskártya), vagy egy olyan dolgot, amivel a felhasználó azonos (például ujjlenyomat). Most csak a jelszavas hitelesítéssel foglalkozunk, azt azonban érdemes megjegyeznünk, hogy igazán biztonságos csak akkor lehet a dolog, ha e három hitelesítési eljárás közül legalább kettőt együttesen és egymástól függetlenül használunk. (A bankautomaták például ilyenek. A pénz felvételéhez a PIN kódra is szükség van. Az egymástól független tárolás jelen esetben azt jelenti, hogy a PIN kód nem a mágneskártyán van tárolva.)

Jelszavas hitelesítés

Ez mind közül a legelterjedtebb, mivel könnyedén megvalósítható, és semmilyen különleges eszközt nem igényel, például mágneskártyát vagy ujjlenyomat-leolvasót. Csupán a felhasználó azonosítóját és jelszavát kell bekérni. A Unix esetében ezt a `login` nevű program végzi, amely a kapott jelszót egyből titkosítja is, majd összehasonlítja a jelszófájlban tárolttal. Ha ez egyezik, akkor beengedi a felhasználót, ha nem, akkor a belépési kérelmet visszautasítja.

A jelszavas azonosításnak akad egy rendkívül nagy hátránya, mégpedig az, hogy könnyű becsapni. A rendszer nem tud különbséget tenni a jogosult felhasználó és azon személy között, aki kitalálta vagy valamilyen módon megtudta a másik jelszavát. Ezért az olyan rendszerekben, ahol a védelmet jelszó segítségével oldják meg, a felhasználókat is komoly felelősség terheli. Egyrészt azért, hogy ne olyan helyre írják fel jelszavukat, ahonnan más is elolvashatja (de az a legjobb, ha le sem írják), másrészt azért, hogy ne könnyen kitalálható jelszót válasszanak maguknak.

Erre a két dologra ugyan már végtelenszer felhívták a felhasználók figyelmét, a többség még mindig egyszerű jelszavakat (például neveket, szótári szavakat) használ. Ezért komoly veszélyt jelenthet az úgynevezett szólistás módszer, amikor a támadó egy listát készít a gyakori jelszavakról, majd ezeket egyenként titkosítja; ezután az egészet összeveti a felhasználók rendszerben tárolt titkosított jelszavaival.

Itt védelmet nyújthat az, ha kikötjük, hogy a felhasználók csak olyan jelszavakat használhatnak, amelyben egyszerre szerepel kis- és nagybetű, illetve szám. Egy másik megoldás lehet a jelszófájl sózása. Ez azt jelenti, hogy a felhasználók jelszavához titkosítás előtt hozzáadunk egy n bites véletlen számot. Ezt az n bites számot is titkosítjuk, majd a jelszófájlban tároljuk. Természetesen, ha a felhasználó jelszót változtat, ez a szám is változik.

A sózás abban nehezíti meg a támadó dolgát, hogy jelentősen megnöveli jelszó-listájának a méretét. Ha például a támadó úgy gondolja, hogy az „almafa” karaktersorozatot valaki jelszóként használhatja, akkor a listában fel kell még tüntetnie az „almafa1”, „almafa2” stb. jelszavakat is. Ez azt jelenti, hogy a lista mérete az eredetinek 2^n-szeresére fog változni. A Unix-rendszerek is használják a sózást, ők az n értékének a 12-t választották. Ez már tekintélyes növekedést jelenthet a jelszófájlban.

A sózásnak elsősorban nem az az érdeme, hogy növeli az amúgy is csak pár megabájt méretű, gyakori jelszavakat tartalmazó lista méretét. Inkább az, hogy a nyers erő (broute force) módszerét alkalmazóknak keseríti meg az életét. Minden jelszó feltörhető, csak idő kérdése. Ha viszont a felhasználó kizárólag kisbetűkből álló, öt karakter hosszúságú jelszót használ, akkor az összes lehetőség végigpróbálása már nem tűnik annyira

kivitelezhetetlennek. A sózás használatával azonban jelentősen növelhetjük ezt az időt, bár az igazi megoldás az lenne, ha a felhasználók kizárólag legalább nyolc, kis- és nagybetűket, illetve különleges jeleket egyaránt tartalmazó jelszavakat használnának.
Az újabb Unixok abban is segítik a felhasználói jelszavak védelmét, hogy nem a mindenki számára olvasható */etc/passwd* állományban tárolják a titkosított jelszavakat, hanem egy, a felhasználóktól elzárt helyen. Azok a programok, amelyek ehhez hozzáférhetnek (például `login`, `passwd`) csak késleltetéssel adják vissza az eredményt, ezzel is lassítva a támadó munkáját.

Vírusok

A biológiaórán megismert vírusok olyan életformák, amelyek rendkívül egyszerű szerkezetűek: csak egyetlen DNS-ből (vagy RNS-ből) és az azt védő fehérjeburokból állnak. A vírusok ezért nem is tekinthetők valódi élőlényeknek, maguktól szaporodni sem képesek. Így hát önmaguk reprodukálása céljából hihetetlenül gonosz tetteket hajtanak végre: egészséges sejteket támadnak meg, és beépítik a saját DNS-üket a sejt élettani folyamatait szervező központjába. Tulajdonképpen átveszik az irányítást, azaz a sejtet arra kényszerítik, hogy mindennapi tevékenysége helyett inkább további vírusok előállításával foglalatoskodjon.
Az informatikaórán megismert vírusok olyan programrészek, amelyeknek a legfontosabb feladatuk az, hogy saját magukat ismételjék. Például más programokat úgy változtassanak meg, hogy azok tartalmazzák magának a vírusnak a kódját is (vagy annak egy másik változatát), ezáltal már a fertőzött program is a vírus újbóli megismétlésével fog foglalatoskodni. A számítógépes vírus működési elve tehát a biológiai víruséra rímel. Ahogy a biológiai vírus sem önálló élőlény, és a szaporodásához szüksége van egy gazdasejtre, úgy a számítógépes vírus sem önálló program, inkább csak gépi kódú utasítások halmaza, amelyeknek a végrehajtásához egy gazdaprogramra van szükség.
A vírussal megfertőzött sejt – miután több ezer új vírust gyártott le – rövid időn belül elpusztul. Vannak azonban olyan vírusok (az úgynevezett csendesen fertőző vírusok), amelyek nem okoznak jelentős károsodást a gazdasejtben, csupán a vírus további szaporodását teszik lehetővé. A számítógépes vírusok az utóbbi elvet követik, mivel lételemük a rejtőzködés. Ha ezt nem tennék, a felhasználó rövid úton felfedezné őket. Ezért egy életképes vírus a megfertőzött programot mindig csak annyira változtatja meg, hogy az a további programok megfertőzése mellett eredeti feladatát is kifogástalanul végezhesse. Ez a gyakorlatban a legtöbbször úgy működik, hogy a vírus kódja a megtámadott futtatható állomány végére kerül, és úgy változtatja meg a fejlécet, hogy először a vírus kódja hajtódjon végre. Miután a vírus elvégezte a dolgát (további állományokat fertőzött meg), az eredeti program is végrehajtásra kerül. Ezek az úgynevezett linkvírusok – ez a legősibb és legelterjedtebb vírusfajta, ám bizonyos alkalmazások létrejöttével megjelent egy, talán még tíz évvel ezelőtt lehetetlennek is mondott új linkvírus alfaj: a makróvírus. Legfontosabb tulajdonsága, hogy adatfájlokkal érkezik. Egy W betűvel kezdődő nevű operációs rendszernek van például egy szintén W-vel induló nevű szövegszerkesztője, amely lehetővé teszi, hogy a dokumentumokban a felhasználó úgynevezett makrókat helyezhessen el. A makró tulajdonképpen egy egyszerű programnyelv, amelyet a szövegszerkesztő értelmez, majd végrehajt. A makró eredetileg minden bizonnyal a „felhasználói beavatkozást igénylő (interactive) dokumentumok" létrehozását szolgálta, de elég fejlett volt ahhoz, hogy például fájlkezelő műveletekre használhassunk. Ennek köszönhetően megfelelő teret kínált vírusok számára, amelyek magát a szövegszerkesztőt használták futási közegként. A makróvírusok megjelenése két dolog miatt jelentett nagy csapást a személyi számítógépek biztonságára nézve: egyrészt azért, mert makróvírus írásához nem volt szükség magas szintű programozási ismeretekre; másrészt azért, mert a dokumentumcserék akkor még teljesen gyanútlanul mentek végbe, se a felhasználók, se a vírusirtók nem számítottak a dokumentumokból előbújó szörnyetegekre. Az internet elterjedése, és a különböző internetes alkalmazások egy rendszerbe történő „összevonása" a makróvírusok szétáramlását még jobban megkönnyítette…
Akadnak olyan vírusok is, amelyek másképp támadnak a rendszerre (bár többségük már csak az emlékezetünkben él). Ilyenek voltak például a társvírusok, amelyek nem nyúlkáltak a gazdaprogram kódjába, hanem az azt tartalmazó futtatható állományt nevezték át és tették rejtetté, majd a helyére saját magukat másolták. A gazdaprogram elindításakor valójában a vírus indult el, amely természetesen elindította az eredeti programot is. Ezek a vírusok azonban hamar lebuktak, ha valaki a gazdaprogram eredeti állományában keresett adatot vagy programkódot (azaz az úgynevezett overlay technikát használták). A társvírusok egyik rendszerre jellemző alfaját az „EXECOM" vírusok képezték, amelyek az MS-DOS-nak azt a tulajdonságát használták ki, hogyha parancssorból történő indításkor a felhasználó nem ad meg kiterjesztést, akkor a rendszer először a ***.COM*** kiterjesztéssel keres állományokat, s ha nem talál, akkor ***.EXE***-vel folytatja, végüla ***.BAT***-tal zárja a próbálgatást. Ha például létezett egy ***valami.exe*** nevű állomány, a vírus mellérakott egy ***valami.com*** nevűt, amely magát a vírust tartalmazta. Ezután már csak abban kellett reménykedni, hogy a felhasználó a programot a `valami` parancs kiadásával indítja el. Az EXECOM vírusok azonban egy Norton Commander nevű alkalmazás elterjedése következtében teljesen kihaltak.
Az eddig említett vírusok nagy hátránya, hogy addig teljesen hatástalanok, amíg egy fertőzött állomány végrehajtásra nem kerül. Ezért vannak olyan vírusok, amelyek már a gép indulásakor, még az operációs rendszer betöltődése előtt szeretnek aktiválódni. Ezek az úgynevezett indítóvírusok (boot virus), amelyek az indítórészben laknak,

így a gépszintű program (firmware software) (PC esetében a BIOS) már a gép indítása után egyből nekik adja a vezérlést. Az indítóvírusok általában egyben vírusok is, tehát nemcsak a hajlékonylemez-meghajtóba tett lemezek indító szektorát, hanem a gépen található alkalmazásokat is megfertőzik. Az indítóvírusok azonban nem jelenthetnek veszélyt az igazi védett módú operációs rendszerekre (például a Linuxra vagy a Windows XP-re).

Rejtőzködés

A vírusok tehát olyan programkódok, amelyek saját magukat sokszorosítják. A vírusírók pedig olyasfélék, akik olyan programkódokat írnak, amik saját magukat sokszorosítják. Ez már önmagában is kihívás, de a vírusírók a szaporodáson kívül mindig szeretnek valamiféle „magasabbrendű” célt kitűzni gyermekeik számára. Ez lehet például adatszerzés, rombolás, vagy éppen valami ártatlan kis semmiség. Mindenesetre minél szaporább egy vírus, annál nagyobb esélye van arra, hogy a kitűzött célt teljesítse.

A vírus tevékenysége azonban mindig valamilyen változással jár. Például, ha megfertőz egy futtatható állományt, akkor annak a mérete megváltozik. Mivel ezek a változások felismerhetők, fennáll a veszély, hogy a felhasználó gyanút fog, ami egyben (legalábbis azon a rendszeren) a vírus pályafutásának a befejezését jelentheti. A korszerűbb vírusok nemcsak arra ügyelnek, hogy a fertőzött programok futtatása közben a felhasználó semmi rendelleneset ne észleljen, hanem arra is, hogy a szaporodással járó változásokat minél jobban eltitkolják. Hogy erre milyen módszereket használhatnak, az nagymértékben függ magától az operációs rendszertől is. Például az MS-DOS, ami tulajdonképpen nem más, mint egy eljáráscsomag, semmiféle korlátozást nem szabott a programok számára. Mi több, a teljes vezérlést az alkalmazás kezébe adja, így a vírusok nyugodtan módosíthatták az indítórészt, vagy magukra irányíthatták a különböző rendszerhívásokat. Ezáltal olyan dolgokra is képesek lehetnek, hogy például a fertőzött állomány méretének a lekérdezésekor hamis eredmények szülessenek (így leplezve a méretváltozást).

Ilyesmi nem lehetséges egy olyan valódi védett módú operációs rendszer esetében, amelyben minden folyamat szigorúan csak a saját memóriaterületével rendelkezik. Ez természetesen nem jelenti azt, hogy például Linuxra nem lehet vírust írni. Vírust minden operációs rendszerre lehet írni, csak nem mindig érdemes. Ha például a vírust alkotója azzal a képességgel ruházza fel, hogy egy ismert biztonsági rést használjon ki, akkor a vírus sok csúnya dologra képes lehet, megfertőzheti magát az operációs rendszert is.

Mindenesetre a többfelhasználós rendszerek nagyobb ellenállást tanúsíthatnak a vírustámadásokkal szemben, hiszen az egyszerű felhasználók a programok többségét csak futtathatják, de nem módosíthatják. Az elindított vírus így csak a felhasználó tulajdonában lévő állományokra jelenthet veszélyt, a rendszer egészére azonban nem. Kivéve, ha nem a rendszergazda indít fertőzött állományt. A makróvírusok azonban a többfelhasználós rendszerekben is életképesek, mivel a dokumentumok a felhasználók tulajdonában vannak.

Vírusok nyomában

Sem most, és valószínűleg a jövőben sem fog létezni olyan algoritmus, amelyik egyértelműen és tévedhetetlenül képes kimutatni bármilyen vírus jelenlétét a rendszerben. Bár többféle módszer létezik, amelynek alapján megsejthetjük, hogy vírustámadás áldozatai lettünk, illetve felismerhetünk már felfedezett vírusokat, de ez is csak abban az esetben lehetséges, ha az adott vírus ügyetlenül vagy egyáltalán nem védekezik az alkalmazott módszer ellen.

A legismertebb ilyen módszer a szignatúrakeresés. Ez arra a feltevésre épül, hogy a vírusok kódjának valamely része biztosan egyedi, azaz nem fordulhat elő olyan állományban, amelyik az adott vírussal nincs megfertőzve. Ez máig a leghatékonyabb vírusészlelő módszer, és a vírust még a működésbe lépése előtt felfedezhetjük. Hatástalan azonban az ismeretlen vírusokkal szemben, továbbá a korszerű polimorf vírusokkal sem tud mit kezdeni, mivel ők képesek a saját kódjuk megváltoztatására. Egy másik nem elhanyagolható dolog, hogy a szignatúra-adatbázist folyamatosan frissíteni kell, és az adatbázis növekedésével a keresés ideje is növekszik.

Mivel a polimorf vírusok könnyedén kijátszhatják a szignatúrakeresőket, a heurisztikus keresés kifejlesztése vált szükségessé. A módszer lényege annyi, hogy a víruskereső elkezdi értelmezni a program kódját, gyanús tevékenységeket keresve benne. (Ilyen gyanús dolgok lehetnek például a különböző titkosító és polimorfikus eljárások). Előnye, hogy az ismeretlen vírusokat is képes felismerni, viszont nagyon nagy a hibaszázalék: nagyszámú a téves riasztás, és nem minden esetben akad a vírus nyomára.

Mivel a vírusok kivétel nélkül változásokat idéznek elő, a változások megfigyelésével felismerhetjük őket. Ehhez az kell, hogy még a vírus nélküli rendszerről mindent fel kell jegyeznünk (például a futtatható állományok tartalmát, illetve annak „lenyomatát” a CRC algoritmus használatával). Ezután már csak rendszeres időközönként össze kell hasonlítani a rendszer jelenlegi állapotát az eredetivel – ha vírus van a rendszerben, akkor a víruskereső észreveszi, de csak azután, miután a vírus működésbe lépett. További gond, hogy változást nem csak vírus okozhat, ezért téves riasztás is előfordulhat.

Láthattuk tehát, hogy mindegyik módszernek megvan az előnye és hátránya, így igazán hatékony megoldást csak együttes használatukkal érhetünk el. A jövő azonban bíztató, mivel kutatások folynak olyan „intelligens” víruskereső programok kifejlesztésére, amelyek képesek csapdába ejteni a gépet megtámadó vírust. Ezt úgy érheti el például, hogy egy pontosan ismert csali mindenfelé programokat helyez el a rendszerben, majd azokba véletlen írásokat kezdeményez, ezáltal „bírja rá” a vírust a csaliprogram megfertőzésére. Mivel ismeri az eredeti csalit, a vírus kódja könnyedén kinyerhető, elemezhető – esetleg önműködően azt észlelő és eltávolító

eljárást is készíthet hozzá. Ez a módszer sem lehet százszázalékosan megbízható, amíg nem lehetünk biztosak abban, hogy a memóriába ugyanazokat az adatokat kapjuk vissza, mint ami a lemezen szerepel (és nem azt, amit egy kósza vírus módosított). Ugyanez a nehézség a többi módszer esetén is fennáll, így az igazi megoldás talán az lenne, ha a víruskereső is az operációs rendszer belsejében foglalna helyet. Ám a siker ekkor sem lenne garantált.

A féreg

Féregvírusnak is szokás nevezni, mivel a férgeket is szaporodásra tervezték. A féreg (worm) azonban nem vírus. Ha megint egy, a biológiaórán megismert szereplőhöz szeretnénk hasonlítgatni, akkor a féreg a baktérium megfelelője lehet. A baktérium egy önálló élő szervezet, amely képes saját magától szaporodni, méghozzá gyorsan – elegendő tápanyag jelenlétében akár exponenciális ütemben is. A féreg tehát olyan önálló program, amelyik saját magát sokszorosítja és indítja el, egészen addig, amíg a rendszer össze nem omlik.
A legtöbb féreg azonban mindig nagyban gondolkozik. Míg a vírusok minél több állomány megfertőzésére törekszenek, addig a férgek az internet segítségével minél több gépre el szeretnének jutni. Mindig „egyedül" utaznak, és amint megérkeznek egy helyre, egyből azon gondolkoznak, hogyan juthatnának el más gépekre.
Az első igazán nagy pusztítást végző féreg 1988-ban szabadult el, és a Berkeley Unixok biztonsági réseit használta ki. Ez a féreg két részből állt: a féreg magjából és egy áthúzó programból. Az utóbbit a megtámadni kívánt gépen kellett lefordítani és lefuttatni. Feladata csak annyi volt, hogy kapcsolódjon ahhoz a géphez, ahonnan érkezett, majd töltse át a féregmagot, majd fordítson és futtasson. A féreg magja megnézte a */etc/hosts* nevű állományt, amelyből kiderült, hogy az adott gép mely további gépekkel áll kapcsolatban. Ezekre áttöltötte az áthúzó programot, és kezdődött minden az elejéről.
Az igazán érdekes azonban az, hogy a féreg miként volt képes rávenni a távoli gépeket az áthúzó program lefuttatására. Három módszert is kipróbált rá. Először az `rsh` (remote shell) parancs segítségével próbálta meg elérni a távoli gép parancssorát. Ha sikerült, akkor az áthúzóprogramot áttöltve és lefordítva megvalósult a fertőzés.
Amennyiben ez nem jött volna be, a `finger` démon egy ismert távoli átmeneti tár túlcsordulási hibáját aknázta ki. Ennek a módszernek az a lényege, hogy értékként egy jó nagy, különlegesen összeállított adatblokkot adunk át. Ez az adatblokk bőven nagyobb volt, mint az átmeneti tár, így az túlcsordult, és a verem felülírásra került. Ha képesek vagyunk távoli program vermébe írni, akkor megváltoztathatjuk annak a visszatérési címét. Így a `finger` feladata befejezte után nem az eredeti helyre tért vissza, hanem elindított egy parancssort.
A féreg ehhez kapcsolódva könnyedén telepíthette magát.
A harmadik próbálkozás a Sendmail nevű levelezőrendszerrel történhet: a féreg egy másik gépre (levélként) átküldhette, majd futtathatta magát.
A hálózatok többségében a felhasználóknak nem csak egy géphez van hozzáférésük, így a féreg arra is vette a fáradságot, hogy nekiálljon feltörni a felhasználók jelszavát, és az ilyen módon megszerzett gyenge jelszavak segítségével megpróbáljon a többi gépbe is beférkőzni.
A vírusokhoz hasonlóan a férgeknek is rejtőzködniük kell, hogy minél tovább szaporíthassák önmagukat. Ezért nem jó, ha a féreg túl korán lelassítja a megfertőzött gépeket. A most tárgyalt féreg például a fertőzés előtt megnézte, hogy hány példányban fut már a kérdéses gépen. Ha már több mint hét példányban futott, akkor inkább más célpont után nézett.
Ez a féreg annyira hatékony volt, hogy rövid időn belül több ezer gépet volt képes megfertőzni és megbénítani (ami akkoriban az internet jelentős részét képezte). A féreg azért is mérföldkőnek számított a számítógépes biztonságtechnika terén, mert képes volt olyan biztonsági hibákat kihasználni, amelyek segítségével felhasználói beavatkozás nélkül tudott terjedni.
Azóta több féreg is átsöpört a világon, volt olyan is, amelyik nem volt ennyire összetett, több közülük csupán egy levelezőügyfél hibáját próbálta kihasználni. Általában azért a „felhasználó" is hibás volt, mert megnyitotta a férget tartalmazó levelet.

Hátsó ajtók és trójai falovak

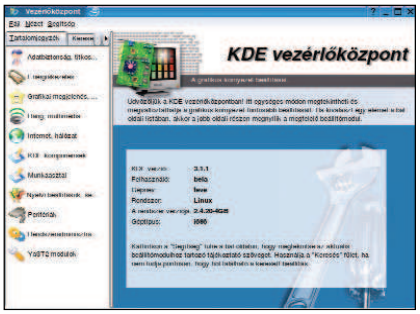
A hátsó ajtó egy olyan kód, amelyik rést nyit a rendszeren, és illetéktelenek belépését teszi lehetővé, illetve adatokat juttat ki a rendszerből. A trójai faló egy olyan alkalmazás, ami valami mást is csinál, mint amit hirdet magáról, és azt a valamit nagyon jól titkolja.
Fontos, hogy sem a hátsó ajtó, sem a trójai faló nem vírus és nem is féreg, mivel egyik sem szaporodik. Az más kérdés, hogy sok vírus egyben hátsó ajtó is, tehát alkotója számára belépést biztosít a fertőzött gépekre. A trójai programok is tartalmazhatnak hátsó ajtót (ebben az esetben a hátsóajtó-készítés a titkos feladat), de lehetnek például vírusgazdák is. A vírusgazda nem egy vírussal fertőzött program, csak tartalmazza a vírust, és végrehajtáskor azt útjára engedi a rendszerben.
A következő részben szó lesz még a rejtett csatornákról és a felhasználói réteg alkalmazásairól. Elsősorban a démonokról írok majd, amelyek fontos építőkövei a Unix-rendszereknek (még ha nem is tekinthetők közvetlenül az operációs rendszer részének). Nem kanyarodunk el azonban véglegesen a biztonság kérdésétől, mivel megtudhatjuk azt is, hogy egy könnyen „korrumpálható" démon milyen mértékben áshatja alá a rendszer biztonságát (a válasz: nagymértékben).

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

Hogyan térjünk át Linuxra lépésről lépésre?

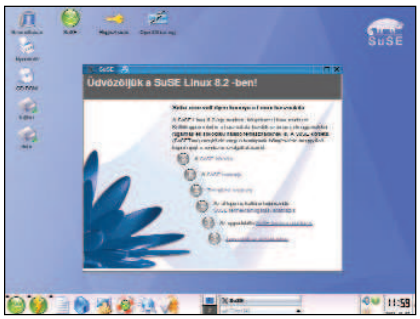
Mostanában egyre több embert foglalkoztat a címbeli kérdés, hiszen egyre több szó esik a szabad programokról mint az új irányzatról, élén a téma összefogójával, a szabad programmozgalom zászlóvivőjével: a Linux operációs rendszerrel. Vajon miért érdeklődnek az emberek e témakör kerengő szóbeszédei iránt?

Hogyha valami újat hall az ember, azonnal érdekelni kezdi, rögtön kíváncsi lesz, és nincs ez másképp akkor sem, amikor az ismerőseim a Linuxról kérdeznek. Egyszer csak eljön a pillanat, hogy engedve a kíváncsiságuknak, megpróbálnak feltenni egy ilyen operációs rendszert a gépükre. Igazából itt kezdődnek a nehézségek, ugyanis a másoktól hallottak alapján próbálnak meg elindulni, ám azok korántsem szilárd talajon álló útmutatók, inkább csak úgy lógnak a levegőben. Nem árt tisztában lenni néhány szakmai kérdéssel, mielőtt ilyen vállalkozásba kezdenénk, de ez általában nem így van, nem csoda hát, hogy a Linux a legtöbb ember számára túl nagy falatnak bizonyul. Ennek természetesen gyomorrontás a vége, és teljesen elmegy a kedvük ettől az új hóborttól. Pedig valójában nem nagy ördöngösség egy ilyen rendszer telepítése és használata, csak épp a hiányos adatok alapján végzett kísérlet gyakran kudarccal jár, és ez a felsülés hajlamos elvenni az emberek kedvét. Ezek elkerülése érdekében ezzel a cikkel egy új sorozat indul a magazinban, amely a kezdő Linux-felhasználókat lépésről lépésre végigvezeti az átállás folyamatán – a telepítéstől egészen az egyes alkalmazási területeken adott lehetőségek kihasználásáig. Abban a reményben kezdek neki cikkem írásának, hogy az olvasóra gyakorolt hatás eredményeképpen sokkal több ember kóstol bele a szabad programok világába, és a fiaskók elkerülésével csökken majd az effajta előítélet, amely a „használhatatlanság” bélyegét sütötte a Linuxra. Ehhez azonban az is kell, hogy aki belefog a telepítésbe, bizonyos szinten azért tisztában legyen néhány fogalommal. Azt tudom mondani, hogy aki telepített már Windows 2000-et, vagy XP-t, az itt is boldogulni fog, de ez nem jelenti azt, hogy egy, a témában járatlan olvasónak ne sikerülhetne, hiszen a bemutatott terjesztés egy átlagos rendszerre 1–2 egérkattintással telepíthető.



Melyiket válasszam?

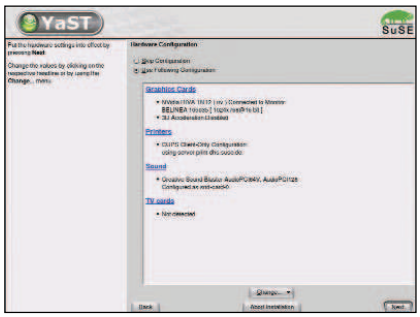
Mindenekelőtt szükségünk lesz egy terjesztésre a telepítéshez, amelyet célszerű úgy megválasztani, hogy később mi, kezdő felhasználók ne szenvedjünk vele sokat. Ezt a tényt szem előtt tartva jelen cikkben a SuSE Linux 8.2 változatának a telepítését szeretném bemutatni. Megjegyzem, hogy a korábbi változatok telepítése is csak alig különbözik a jelenlegitől, ezért ha valaki az itt leírtakat egy számára elérhetőbb 8.1-es vagy 8.0-s változatra próbálja meg alkalmazni, az nagy valószínűséggel működni fog.
Itt szeretném megjegyezni, hogy hamarosan a boltokba kerül a 9.0-s változat, amely a tapasztalatok alapján a telepítést tekintve szinte egyáltalán nem fog különbözni az itt leírtaktól (legfeljebb még egyszerűbb lesz), így ha valaki a cikket olvasva kedvet kap, és szeretne magának vásárolni egy SuSE Linuxot, annak talán érdemes megvárni az új, feltehetően szebb és jobb változatot, mielőtt megvenné a mostanit, aztán vágyna az újabb után.
Ami az erőforrásokat illeti: ha rendeltetésszerűen szeretnénk használni a gépet, ne adjuk alább egy 300 MHz-es Pentium II Celeron processzornál és 128 MB memóriánál. Ezenkívül szükség lesz legalább 2 GB szabad lemezterületre, de ha minden csomagot fel szeretnénk tenni, az 5–6 GB-ra is rúghat – amely azonban végletes eset, átlagosan legfeljebb 2,5 GB, de már ebben az esetben is egy jól megrakott rendszer birtokában vagyunk.

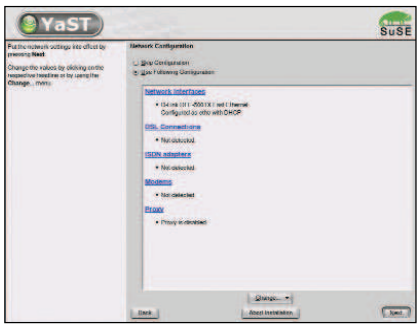


Telepítés

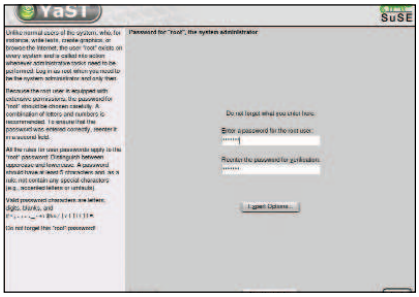
A SuSE telepítésének alapvetően két módja lehetséges: hálózatról vagy CD-ről történhet. A hálózat hátránya, hogy nem árt, ha minél gyorsabb kapcsolat áll rendelkezésünkre, a programok azonban rengeteg FTP-kiszolgálón teljesen ingyen hozzáférhetők. A CD-s telepítéshez még internetkapcsolatra sincs szükségünk, viszont a CD-állományok sehonnan nem tölthetők le, ehhez meg kell vennünk a SuSE termékét, amely pénzbe kerül. Ám ne ijedjünk meg, a dobozban ugyanis nemcsak CD-k (DVD-k) vannak, hanem két igen vastag magyar nyelvű kézikönyv, amely főként a Linuxot még soha nem látott ember életét könnyítheti meg jelentősen, ezen felül három hónapos ingyenes telepítési támogatást is nyerünk.
A hálózatról történő telepítés során szükségünk lesz indító CD-re vagy hajlékonylemezre, amit az ➲ ftp://ftp.suselinux.org/pub/suse/boot/ könyvtárban találunk. A lemezlenyomatokat letöltve vagy CD-re írva

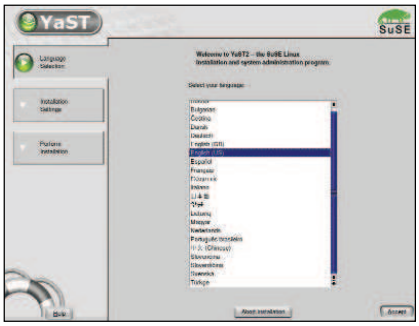
(*boot.iso*), vagy több hajlékonylemezre másolva a `rawrite` program segítségével megkapjuk a kívánt indítólemezeket. A rendszerindítás után a hálózati kapcsolat megadását követően a megfelelő helyen ki kell választanunk az adatforrást, amelyről az ➲ ftp.suselinux.hu kiszolgálón lévő ide vonatkozó leírásból tájékozódhatunk.
Nézzük azonban most az egyszerűbb telepítési módot, ahol az első néhány lépés elhagyható. Tegyük be a program





első korongját vagy a DVD-jét, és indítsuk róla a gépet. Amikor bejön az indítóképernyő, nyomjunk meg a le–fel billentyűket, hogy tájékozódni tudjunk. A képernyő alján látszik a felbontás. Ha ez nekünk nem felel meg, az F2 billentyű segítségével megváltoztathatjuk. A telepítés során a grafikus kártyánk VESA-módban üzemel, amely 60 Hz-es képfrissítési frekvenciát használ, ennek fényében kell tehát a monitorunknak megfelelő felbontást kiválasztani. Ha nem vagyunk biztosak benne, válasszuk a 640×480-as üzemmódot, az biztosan működik.
Ha most kiválasztjuk az ***Installation*** menüpontot, elindul a telepítés folyamata. Pár másodperc elteltével megkapjuk a telepítőfelületet, elsőként a nyelvválasztási lehetőséggel. Javasolt a magyar nyelv kiválasztása a későbbi esetleges nehézségek elkerülése végett.
Ha ez megvan, a következő lépésre kattintva a telepítő előkészíti a rendszert, majd megmutatja a tervezett telepítési menetet, és lehetővé teszi számunkra, hogy bármelyik javaslatot megváltoztassuk. A képernyőn található egyes részlegek a telepítés főbb jellemzőit tükrözik, a címek alatt kisebb betűvel szedve található az egyes elemek részleteire vonatkozó tájékoztatás.
Ha át akarjuk őket állítani, a címsorra kattintva új paneleket kapunk. Az új felületet az alábbiak szerint kell értelmezni az egész telepítés folyamán: bal oldalon található egy általános terület, ahol mindig a pillanatnyi lépéshez tartozó magyar nyelvű segédletet találjuk, ebben röviden ismertetik az adott folyamat lényegét, tippeket adnak a beállí-





tásra, amiket természetesen ajánlott betartani. A képernyő nagyobbik felén található a beavatkozófelület, itt végezhetjük el a telepítés folyamán a beállításokat.

Az észlelt beállítások

A billentyűzetre és egérre vonatkozó adatok az esetek túlnyomó többségében helyesek (értsd: én még nem találkoztam olyannal, hogy rosszul ismerte volna fel), ezzel nem kell foglalkoznunk, sőt ha tényleg teljesen kezdő felhasználók vagyunk, semmi egyébbel se törődjünk, csak folytassuk a következő telepítési lépéssel. Ha azonban egy-két beállítást meg szeretnénk változtatni, akkor kalandozzunk el egy kicsit ezen a képernyőn.

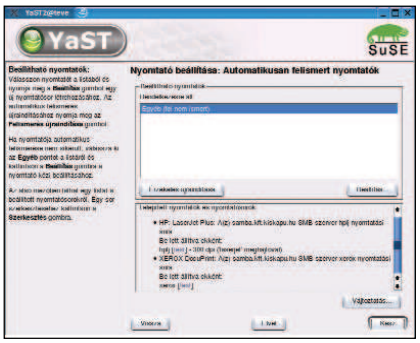
Lemezfelosztás

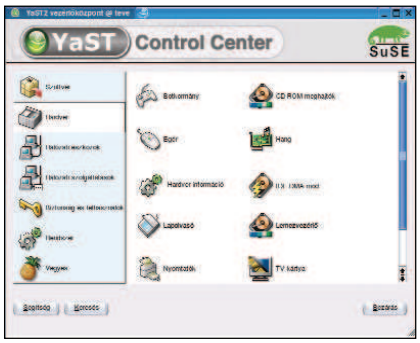
Az első fontos rész a lemezterület felosztása, amelyben a Linux számára szükséges lemezrészek kialakítását végezhetjük. Kevesen tudják, hogy minden operációs rendszernek szüksége van valamilyen lemezrészre ahhoz, hogy használni tudjuk. A Linux ezt megoldja magának, keres egy elegendő méretű szabad lemezrészt, és a maga igényei szerint lefoglalja, megteszi a szükséges beállításokat. Ha nincs ilyen felosztható terület, akkor megpróbál átméretezni egy olyan lemezrészt, amin elég szabad hely található, hogy a számára elegendő területet fel tudja szabadítani. Célszerű a telepítés megkezdése előtt előkészíteni a merevlemez egy részét, amin nincs semmi, de ha ezt nem tudjuk megoldani, hagyjunk valamelyik meghajtón 5 GB körüli szabad helyet, és a többit bízzuk a telepítőre. Most egy kicsit azokhoz szólnék, akik jártasak a lemezfelosztás művészetében. A Linux számára főként két elsődleges vagy kiterjesztett lemezrész szükséges. Az egyik egy, a fizikai memóriával arányos méretű csereterület (swap), ahová a rendszer a ritkán használt memóriaterületek tartalmát helyezi át – felfoghatjuk virtuális memóriának is. A másik területen helyezkedik el maga az operációs rendszer és a felhasználók állományai. A csereterület lemezrészének létrehozásánál a méret megadása után válasszuk ki a csereterület-fájlrendszert a bal alsó lenyíló menüből, majd mentsük a változtatásokat. A fő (gyökér) lemezrész létrehozásához foglaljunk le körülbelül 5 GB területet, és javasolt a ReiserFS fájlrendszer használata, amelyet szintén a bal alsó lenyíló menüben találunk.
A befűzési pontok mutatják meg, hogy a rendszer hová fűzze az egyes lemezrészeket, vagyis a használat során hol látszódjanak. A két alaptípus létrehozásánál a lemezkezelő program önműködően beállítja ezt nekünk, de ha nem így volna, mert más módon kezdtük el a létrehozást, akkor a csereterület-lemezrész csatlakoztatási pontja `swap` legyen, a fő lemezrészé pedig `/`. Ezek után minden adott, hogy megkezdhessük a telepítést.
Tipp: ha nem boldogulnánk a telepítő akaratos átméretezési vágyával, válasszuk a lemezkezelő panel jobb alsó sarkában található ***Szakértő*** menü ***Partíciós tábla újraolvasása*** menüpontját, ami után a telepítés előtti állapotból indulhatunk ki. Az egész folyamat nem bonyolultabb, mint egy Windows XP telepítése során történő lemezkialakítás, csak ne ijedjünk meg tőle – ugyanazt kell tenni, csak épp NTFS fájlrendszer helyett itt a ReiserFS-t kell kiválasztani, és szükségünk van még egy kisméretű csereterület típusú lemezrészre is. Újra hangsúlyoznám azonban, hogy aki az

itt leírtakat nem érzi a magáénak, nyugodtan folytassa a telepítést a felajánlott beállításokkal, a Linux úgyis fel fog menni a gépre.
Kényelmes szolgáltatás az is, hogy a már létező windowsos, DOS-os és egyéb lemezrészeket önműködően hozzá fogja fűzni a rendszerhez, így minden előkészítés és fájdalom nélkül írhatjuk, olvashatjuk a többi lemezen lévő fájlokat. Ezenkívül lehetőségünk adódik a gépen található régebbi operációs rendszerek használatára is, mivel a SuSE ezt is minden külső beavatkozás nélkül felismeri, és beteszi a rendszertöltő menübe, ahol minden indításkor lehetőségünk lesz választani.
A többi lehetőség megváltoztatására gyakorlatilag semmilyen átlagos számítógép esetén nincs szükség. Egyetlen említésre méltó pont az időzóna beállításainál található óra beállítása lehetőség, amelyet nem célszerű UTC-re állítani, mégha azt is akarja, válasszuk inkább a helyi idő szerinti beállítást. Mindenképp tegyünk így, ha mellette más operációs rendszert is használunk, mert ezáltal mindegyikben ugyanazt az időt látjuk, más esetben előfordulhat, hogy a Linuxban helyes idő Windows alatt greenwichi idővé válik.

Az összetevők telepítése

Még a telepítésre kerülő programok listáját sem érdemes módosítani. Egyrészt azért, mert nem tudjuk még, hogy mi mire való, másrészt mindezt később ugyanilyen kényelmes formában megtehetjük, és jelen esetben talán jobb, ha most arra törekszünk, hogy felépítsünk egy alaprendszert, amellyel aztán megkezdhetjük az ismerkedést.
Ha készen vagyunk, a telepítés következő szakasza a fájlok másolása és kicsomagolása a merevlemezre, közben majd kérni fogja az egyes CD-ket, ha épp nem DVD-ről telepítünk.
Az alapfájlok másolása után a rendszer újraindul, ám a telepítés itt még nem fejeződik be. Egyrészt tovább zajlik majd a fájlok másolása és kicsomagolása, másrészt ezután adódik majd lehetőségünk az eszközök beállítására. Előbb azonban a fájlok másolásának végeztével a rendszer kéri a rendszergazda jelszavát. Hogy ezt megértsük, nézzünk egy kicsit a dolgok mögé! A Linux többfelhasználós operációs rendszer, ami azt jelenti, hogy egyszerre egy vagy több gépről többen is használhatják a masinánkat mindenféle nehézség nélkül. Ahhoz azonban, hogy az ebből adódó nehézségeket elkerüljük (például





egymás fájljait ne lássuk, akarva akaratlan ne tudjunk kárt tenni más munkájában és tönkre se tehessük a rendszert), azonosítani kell magunkat a Linux számára. Linuxunk alapesetben ismer minket, és a megfelelő minta egyeztetése után elkönyveli, hogy mi használjuk, és megengedi az ismeretségnek megfelelő műveletek elvégzését. A rendszerben létezik egy olyan alapértelmezett felhasználó, aki a rendszerért felelős, aki megismerteti a rendszerrel, hogy kik használhatják, és aki a gép „üzemeltetését" végzi, ennélfogva bármit megtehet, és mindent lát. Ez a felhasználó a rendszergazda (root). Jelen esetben a mi gépünket mi működtetjük, úgyhogy egy személyben rendszergazdák is vagyunk. Adjunk meg tehát egy jelszót, hogy mások ne birizgálhassák a fontos beállításokat. Később látni fogjuk, hogy leszünk mi még felhasználók is, ehhez azonban még nem érkeztünk el, inkább folytassuk a telepítést.

A hálózati eszközök beállítása

Most következik a hálózati eszközök beállítása. Egyszerű modem, ADSL kapcsolat, hálózati kártya beállítása nem jelenthet gondot, erre itt külön nem is térnék ki, hiszen az eszközt a rendszer valószínűleg felismeri (nagyon ritka darabnak kell lennie, hogy ez ne így történjen), a beállítás pedig ugyanúgy zajlik, mint a többi operációs rendszer alatt. Minden üres mező értelemszerűen van elnevezve, tehát aszerint is kell őket kitölteni! Ezentúl egészen pontos segítséget kaphatunk a bal oldali sávban lévő súgótól.
És most térjünk rá a nehezebb feladatokra: ha olcsó PCI-os WinModemet (softmodem) használunk, ne csodálkozzunk, ha nem ismeri fel a telepítő, ugyanis ez igazából nem is modem. Végleg azért ne temessük el a dolgot, akad némi WinModem-támogatás a Linuxhoz, de ez nagymértékben attól függ, hogy milyen típusú az eszköz. Általában, ha a gyártó nem írt hozzá vezérlőt, akkor nem fogjuk tudni használni – ilyen esetekben a SuSE is azt javasolja, hogy vegyünk másikat. Ha tehát ilyen modemet birtoklunk, egyelőre hagyjuk a beállítását, majd a telepítés végeztével, vezérlőprogramokkal a tarsolyunkban próbáljuk meg beállítani.

Az első felhasználó létrehozása

Ha továbbmegyünk, lehetőséget kapunk az első felhasználó létrehozására. Ez is a fent említett többfelhasználós rendszer következménye. Ismertessük meg magunkat a rendszerünkkel, hogy később beengedjen bennünket. Írjuk be a nevünket, majd adjuk meg a bejelentkezési becenevünket és a jelszavunkat. Később, amikor a rendszer arra szólít fel bennünket, hogy azonosítsuk magunkat (jelentkezzünk be), ezt a becenév–jelszó párost kell majd megadnunk. Fontos, hogy ne rendszergazdaként jelentkezzük be, ugyanis ha így használjuk a rendszerünket, az számos veszélyt rejt magában: például végzetesen elronthatjuk némelyik beállítást, hiszen mindenhez jogunk van.

Az eszközök beállítása

A felhasználó létrehozása után következzen az eszközök beállítása, és ha már ideértünk, álljunk is meg egy kicsit. A telepítő az összes létező összetevőt megpróbálja felismerni, és az esetek többségében sikerül is az összeset eltalálnia. Az eszközellenőrzés után egy hasonló működésű panelt kapunk, mint az elején, a telepítés előkészítésénél. Az egyes összetevőcsoportok címkéi alatt találjuk az abba a csoportba tartozó felismert eszközöket (a videokártyát és a monitort, a nyomtatót, a hangkártyát és a tévékártyát). A videokártya beállításai szinte mindig helyesek, a monitort azonban már nehezebben ismeri fel, ha háromévesnél öregebb. Ehhez találunk néhány szabványos vezérlőt, amellyel használhatjuk a monitort, de ha nagyon értünk hozzá, beállíthatjuk a függőleges-vízszintes eltérítési frekvenciákat, hozzáadhatjuk a kívánt felbontást, amivel

máris megoldottuk a képernyő-információs fájl hiányát. Itt csak annyi megjegyzést szeretnék tenni, hogy ha jól is ismeri fel a telepítő a monitort, az elérhető legnagyobb felbontást igyekszik beállítani. Ez nem mindig előnyös, és olyan módon védekezhetünk ellene, hogy a grafikus kártya panelén belüli ***Színek/felbontások*** menüre kattintva a bejövő ***Munkaasztal tulajdonságai*** ablak ***Felbontások*** fülében csak azt a felbontást hagyjuk meg, amit használni szeretnénk. Később, amikor a ***Vezérlőközpont*** segítségével át akarjuk állítani, ugyanezen a módon megtehetjük. Befejezéskor a ***Teszt*** gombbal ellenőrizzük le, hogy a beállított felbontás és a monitor együtt képes-e működni, látjuk-e a képet. A kipróbálás során lehetőség nyílik némi kalibrálásra is, amellyel a megjelenő kép méretét és pozícióját pontosíthatjuk. A ***Mentés*** gombra kattintva a következő indítás során már az új jellemzők lépnek érvénybe. Ha a próba során mégsem látszódna a kép, egy kis idő elteltével önműködően, vagy a CTRL+ALT+BACKSPACE billentyűvel azonnal visszakapjuk a beállítópanelt, amelyben célszerű vagy a felbontást, vagy a képfrissítési frekvenciákat visszábbvenni, majd újra ellenőrizni, hogy működik-e.
Ami a nyomtatót illeti: ha nem ismerné fel a párhuzamos vagy USB-s nyomtatónkat, ne ijedjünk meg, elképzelhető, hogy az adott típus ellen van némi kifogása. Ilyen esetekben azt tehetjük, hogy valamilyen általános nyomtatóprogrammal hajtjuk meg. Ehhez ritka típusok esetén jól kell ismernünk nyomtatónkat, ám a legtöbb esetben megteszi, ha tudjuk, milyen is valójában. Ha például van egy HP LaserJet 3150-es nyomtatónk, akkor nyugodtan használhatjuk a HP LaserJet-meghajtót, hibátlanul fog működni (ez igaz a DeskJet sorozat tagjaira is). Ezzel azonban most ne foglalkozzunk, majd a telepítés végeztével a YaST program segítségével beállítjuk.

Ami végezetül ránk vár…

Továbblépkedve a ***Hardverbeállítások*** fülről, némi képernyőváltás után értesítést kapunk arról, hogy befejeződött a telepítés, birtokba vehetjük a számítógépet. Ehhez viszont be kell jelentkeznünk, azaz be kell mutatkoznunk a rendszernek. Most kell tehát beírni azt a bizonyos becenév–jelszó párost, amelyet a telepítés során megadtunk. Az ***OK*** gombra kattintva elindul a grafikus felhasználói felület. Ha ez megtörtént, gyakorlatilag készen vagyunk, ám itt még ne álljunk meg, hogy azok is teljes egészében élvezhessék az új rendszert, akiknél hiányosságok léptek fel a telepítés folyamán. Ehhez mindenekelőtt két nagyon fontos rendszerösszetevőt kell megismernünk: a KDE ***Vezérlőközpontot*** és a YaST2 ***Vezérlőközpontot***. Az előbbivel inkább a grafikus felhasználói felület kinézetét, alapvető működési beállításait, a felülettel együtt járó programok beállításait változtathatjuk meg, míg a másodikkal az eszközbeállításokat, a hálózati eszközöket, a hálózati szolgáltatásokat, a rendszerbeállításokat módosíthatjuk, és programokat telepíthetünk vagy távolíthatunk el a rendszerből. Mindezeknek a szolgáltatásoknak az egyszerű eléréséhez a tálcán található a SuSE munkamenü. A menü legelső pontja egy almenü, ezen keresztül juthatunk el az egyes rendszerbeállítások elvégzéséig vagy magáig a YaST2 vezérlőközpontig.

Ismét egy kis nyomtatóbeállítás

Nézzük először a nyomtatónk telepítését, mert mondjuk a rendszer nem ismerte fel, ugyanis hálózati nyomtatót szeretnénk használni. Ehhez válasszuk a ***SuSE munkamenü/YaST menü/YaST modulok/Hardver/Nyomtatók*** menüpontját, az előugró ablakban adjuk meg a rendszergazda jelszavát (mert ez olyan feladat, amit csak a rendszergazda tehet meg), várjuk meg az önműködő felismerést, és vagy fogadjuk el az érzékelt nyomtatókat, vagy ha nem szeretnénk, akkor kattintsunk a ***Beállítás*** gombra, és válasszuk ki, hogy milyen típusú kapura csatlakozik a nyomtató. A következő lépésben meg kell adnunk magát a kaput, de a legtöbb esetben csak el kell fogadnunk az általa felajánlottat. Ezután jön a nyomtató típusának megadása, és itt már ugyanazok az irányelvek érvényesek, amelyekről a telepítés során történő nyomtatóbeállításnál is szó volt. Ha semmilyen módon nem tudjuk működésre bírni a nyomtatónkat, célszerű a gyártó honlapján utánanézni, hogy milyen támogatást nyújtanak.
Ha netán hálózati nyomtatónk van, akkor válasszuk a ***További kapcsolatok megjelenítése*** lehetőséget, majd a megjelenő panelben válasszunk protokollt, ezután tegyük meg protokollfüggő beállításainkat, és újfent válasszunk nyomtatótípust ama bizonyos irányelvek alkalmazásával. Kényelmes szolgáltatás, hogy windowsos hálózaton történő nyomtatótelepítésnél mind a nyomtatókiszolgálót, mind az ott található megosztott nyomtatókat böngészhetjük, semmit sem kell fejből tudnunk a telepítéshez.

Egyéb eszközbeállítások

Ha valamilyen okból nem tudtuk a telepítés során beállítani, hogy milyen hálózati kapcsolattal rendelkezünk, mert például a külső modemünk ki volt kapcsolva, akkor ezen úgy segíthetünk, hogy a ***SuSE munkamenü/YaST menü/YaST modulok/Hálózati eszközök*** menü megfelelő elemének kiválasztásával a nyomtató hozzáadásához hasonlatosan telepítjük. Az első lépés itt is a felismert eszközök elfogadása vagy új eszköz kiválasztása egy listából, majd be kell állítani az adott eszközt, beleértve a kapcsolatra jellemző adatok (felhasználónév, jelszó stb.) megadását is. Fontos megjegyeznünk, hogy itt adódik lehetőségünk az eddigi eszközök beállításainak a megváltoztatására. Alapvetően az összes összetevőre jellemző ez a megoldás: a panel felső részében az újonnan felismert, de még be nem állított eszközök láthatók, alul pedig a már eddig telepített összetevőket sorolja fel a rendszer. Minden esetben ide térhetünk tehát vissza, ha bármi gond adódna.

Végszó

Ha idáig eljutottunk, a grafikus felhasználói felület beépített programjaival a legtöbb általános szolgáltatást már elvégezhetjük: böngészhetünk a neten, leveleket olvashatunk, játszhatunk, és legfőképp nézelődhetünk, illetve ismerkedhetünk a rendszerünkkel, amely filozófiájában esetenként eltér az eddig megszokottaktól, de semmivel sem rosszabb náluk. Fontos megértenünk, hogy csakis fokozatosan, lassan-lassan tapasztalva tudunk előrehaladni, ilyen módon el tudjuk kerülni a közben ért kudarcokat, ami visszavethetne bennünket a fejlődésben. A sorozat további cikkeiben megpróbáljuk részfeladatonként bemutatni a rendszert, kezdve a ***Vezérlőközpont*** és a YaST bemutatásával, valamint a rendszer alapvető szolgáltatásaival és beállításaikkal. Addig pedig mindenkinek csak azt tudom javasolni, hogy ismerkedjen a feltelepített Linuxával, próbálgassa; ne aggódjon, felhasználóként olyan nagyon nem tudja elállítgatni, de ha mégis úgy adódna, egy bátor újratelepítéssel nemcsak orvosolja a hibát, de gyakorlatot is szerez az ilyen jellegű rendszerek üzemeltetésében.

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

Saját IRCBot Perl nyelven

Napi több óra csevegés közben jól jöhet egy társ, aki a rendszeres, könnyen önműködővé tehető feladatokat ellátja helyettünk.

A botokról röptében annyit, hogy a csevegőként futó programok több egyszerű feladatot láthatnak el: a csevegőszoba foglalását, a elárasztás (flood) elleni védelmet, önműködő kirúgást (kick), illetve opot (az op biztonságos jogokkal rendelkező IRC felhasználó); hivatkozásokat és egyéb adatok megjegyzését stb. Mindehhez emberi beavatkozás nem szükséges, esetleg futás közben befolyásolható a működése. A cikkben egy IRC-re (Internet Relay Chat) írt botot mutatok be. A magyar IRCNeten csak a ***sote.irc.hu*** és a ***hub.irc.hu*** kiszolgálókon szabad botokat futtatni. Arról, hogy az adott IRC-kiszolgálón futtathatunk-e botot, általában a MOTD (Message of the day) tájékoztat. Percdíjas internet esetén érdemes saját IRC-kiszolgálót telepíteni, és a fejlesztés ideje alatt azon próbálgatni a botot. Az IRC-ről további adatokat, az alapfogalmakról leírásokat, illetve az illemtanról a ➲ http://www.irc.hu címen olvashatunk.
Perl nyelven írt botom felépítéséhez a `Net::IRC` (v0.73-2) modult használtam. E csomag fejlesztői felületét fogom ismertetni, természetesen nem kimerítően, csupán amennyire e cikk keretei engedik és amennyi ismeret az induláshoz elengedhetetlen. A csomag négy fő összetevőből áll: `Net::IRC`, `Net::IRC::Connection`, `Net::IRC::Event`, `Net::IRC::DCC`. Az utóbbi bemutatására nem térek ki részletesebben, mert nem feltétlenül szükséges a bot működéséhez, és az előzőek ismeretében könnyen megtanulható. A `Net::IRC` modul adja a keretet a bot működéséhez. Eseményvezérelt felépítésének köszönhetően nem szükséges az IRC-protokollt mélységeiben ismerni, és a kiszolgáló által küldött üzeneteket ellenőrizni, hanem az előre meghatározott eseményekre (connect, join, msg stb.) mindössze egy-egy eljárást kell írni, amelyek önműködően meghívódnak. Ezeket a függvényeket önkényesen hozzárendelhetjük az egyes eseményekhez, de nem is mindegyikhez kötelező. A későbbiek folyamán részletezem, hogy ezt milyen utasításokkal lehet megtenni.

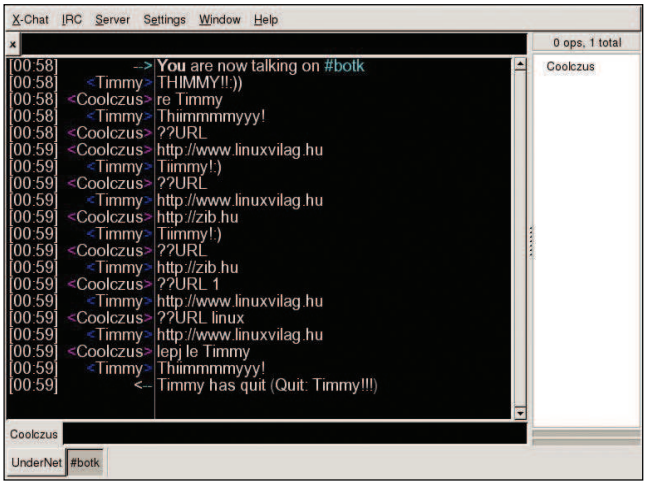
Telepítés

A csomag telepítéséhez Debian GNU/Linux alatt egyszerűen adjuk ki a következő utasítást:

```
apt-get install libnet-irc-perl
```

Ezenkívül szükségünk lesz még a Perl 5.6.0-16-os vagy ennél frissebb változatára is. Más terjesztés esetén is nagy valószínűséggel megtalálható a csomagban, vagy a ➲ http://www.cpan.org weblapról, a ➲ http://search.cpan.org/author/JMUHLICH/Net-IRC-0.74/IRC.pm címről tölthető le. További útmutatást a súgóoldalain, illetve a leírásában (Debian: ***/usr/share/doc/libnet-irc-perl***) lehet találni, sőt a könnyebb megértés kedvéért még egy egyszerű példaprogramot is tanulmányozhatunk.
Az IRC-kiszolgáló csomagból történő telepítése nagyon egyszerű feladat:

```
apt-get install ircd
```



Ezeken kívül választhatjuk még a `dancer-ircd` és a `dancer-services` csomagokat is. Annyival tudnak többet az egyszerűbb IRC-kiszolgálónál, hogy különböző szolgáltatásokat nyújtanak a csevegők számára: csevegőszoba, név (nick) bejegyzése stb. Forrásból való telepítése sokkal bonyolultabb feladat – ennek ismertetése túllép e cikk keretein, de a bátrabbak megpróbálkozhatnak vele.

Első lépések

Az *1. listán* egy nagyon egyszerű IRC bot (Timmy) forrása látható (ez egyébként a CD-mellékleten is megtalálható, az 53. CD Magazin/IRCbot könyvtárában). Alapvetően nem csinál mást, mint kapcsolódik egy IRC-kiszolgálóhoz, belép egy csatornára, és időnként azt mondja, hogy Timmy (a hasonlóság a South Park egyik szereplőjével nem a véletlen műve.) Most tekintsük át lépésről lépésre, hogy melyik utasítás mire való.
A `$DEBUG` változót a hibakeresés megkönnyítésére használom. Futás közben különböző üzeneteket ír ki. Ha a hibaelhárítás be van kapcsolva, a `debug()` függvény az első értékként megadott üzenetet írja ki.
Ezek után létrehozok egy IRC-objektumot, ami ekkor még nem működik, azaz nem kapcsolódik sehova. Ahhoz, hogy a bot elinduljon, a `start()` függvényt kell meghívni, ahogy a forrás legvégén is látni lehet. Az IRC-objektum számára egy kapcsolatot is létrehozok. Megadott értékei, gondolom, nem szorulnak bővebb magyarázatra: `nick`, `server` stb. A `Port` mezőt nem kötelező megadni, alapbeállításként a 6667-et használja. Ha a bot jelszóval védett IRC-kiszolgálóra vagy BNC-re fog csatlakozni, még a `Password` mező is megadható. Természetesen egy IRC-objektumhoz több kapcsolat is létrehozható, de a leírás szerint egyetlen kapcsolattal működik teljesen megbízhatóan. Mire a program eléri az 1.0-s változatot, valószínűleg ezt a gyermekbetegségét is kinövi.
Nem fontos minden kapcsolatot a bot indítása előtt megadni, futás közben (a `start()` függvény meghívása után) is dinami-

1. lista

```
#!/usr/bin/perl -w
use Net::IRC;

# Hibakereses ki/bekapcsolasahoz egy valtozo
my $DEBUG = 1;
sub debug{
      my $msg = shift;
      if ( $DEBUG ){
            print "DEBUG: " . $msg . "\n";
      }
}

# IRC objektum letrehozasa
my $irc = new Net::IRC;

# Kapcsolat objektum letrehozasa.
my $conn = $irc->newconn(
      Server       => '10.0.1.254',
      Port         => '6667',
      Nick         => 'Timmy',
      Ircname      => 'Timmy',
      Username     => 'Timmy'
) or die "Nem lehet kapcsolodni az IRC
      ➥szerverhez.\n";
$conn->{channel} = '#botk';
sub on_connect {
      # Elso argumentumot kiszedjuk.
      my $conn = shift;
      &debug( "Connected..." );
      # Belep a csatszobaba.
      $conn->join($conn->{channel});
      $conn->{connected} = 1;
}
sub on_disconnect {
      # Mindket argumentum kiszedese.
      my ($conn, $event) = @_;
      # Ha megszakadt a kapcsolat ujra
      # csatlakozik.
      &debug( "Disconnected from " .
      ➥$event->from() . " (" .
      (($event->args())[0]) . ").
      ➥Attempting to reconnect...\n");
            $conn->connect();
      }
sub on_join {
      # Mindket argumentum kiszedese.
      my ($conn, $event) = @_;
      # Annak a neve, aki belepett.
      my $nick = $event->{nick};
      &debug( "$nick joined "
      ➥.$conn->{channel}. "." );
      # Ha belepett valaki, akkor Timmy
      # elkialtja magat.
      $conn->privmsg($conn->{channel},
      ➥"THIMMY!!:))");
}
sub on_part {
      # Mindket argumentum kiszedese.
      my ($conn, $event) = @_;
      # Annak a neve, aki kilepett.
      my $nick = $event->{nick};
      &debug( "$nick leaved "
      ➥.$conn->{channel}. "." );
      $conn->privmsg($conn->{channel},
      ➥"Timmmy!!:(");
}
sub on_msg {
      # Mindket argumentum kiszedese.
      my ($conn, $event) = @_;
      # Annak a neve, aki kuldte
      # az uzenetet.
      my $nick = $event->{nick};
      # Az uzenet szovege.
      my $text = $event->{args}[0];
      &debug( "<$nick> $text" );
      # Barmit irnak Timmy mindig ugyanazt
      # valaszolja vissza.
      $conn->privmsg( $nick,
      ➥"Thiimmyyyy!" );
} # on_msg vege
sub on_public{
      # Mindket argumentum kiszedese.
      my ($conn, $event) = @_;
      # Az uzenet kuldoje.
      my $nick = $event->{nick};
      # Maga az uzenet.
      my $text = $event->{args}[0];
      &debug( "<$nick> $text" );
      # most megnezzuk, hogy neki lett
      # e kuldve az uzenet
      if( $text =~ /Timmy/i ){
            # Ha neki jott az uzenet
            # akkor beszol.
            $conn->privmsg
            ➥($conn->{channel},
            ➥"Thiimmmmyyy!");
            # Ha azt irjak neki lepjel le,
            # akkor lelep.
            if ($text =~ /lepj le/){
                  $conn->quit("Timmy!!!");
                  # ezzel lep ki
                  exit 0;
            }
      } # neki szolas vege
} # sub on_public vege

# Esmenykezelok belalitasa.
$conn->add_handler('msg', \&on_msg);
$conn->add_handler('public', \&on_public);
$conn->add_handler('join', \&on_join);
$conn->add_handler('part', \&on_part);
$conn->add_handler('376', \&on_connect);
$conn->add_global_handler('disconnect',
      ➥↓\&on_disconnect);

# IRCBot inditasa.
&debug( "Bot started..." );
$irc->start();
```

IRC-parancsok

- Kapcsolódás a kiszolgálóhoz:
  `/server <szervernev> [port]`
  A kapu értéke elhagyható. Alapértelmezés szerint 6667.
  Például: */server extra.irc.hu*
- Belépés egy szobába: `/join #szobanev`
  Ha az adott szoba nem létezik, akkor létrejön, és mi leszünk az opok.
- Kilépés a szobából: `/part`
- Kilépés a kiszolgálóról: `/quit [uzenet]`
  Ha nem hagyjuk el az üzenetet, kilépéskor kiírja. Ha elhagyjuk, akkor nem ír semmit, illetve néhány IRC-ügyfél (bitchx) véletlen üzeneteket ír be helyettünk.
- Üzenet küldése egy csevegőnek: `/msg <csevego neve>`
- Téma beállítása: `/topic <szoveg>`
  A beállításhoz opra van szükség.
- Op adása: `/op <nick>`
- Kirúgás egy szobából: `/kick <szoba> <nick>`
- Kitiltás egy szobából: `/ban <nev>`
  Itt a név egészen összetett lehet: *nick!ident@hostname*
  Nem kötelező minden részét megadni. Ha például a „valaki” nevű egyént akarjuk kitiltani: `valaki!*@*`
  Ezeket a parancsokat természetesen nem kell minden esetben ismernünk. A legtöbb ügyfél (mIRC, XChat) – elsősorban a grafikusak – esetén gombokkal is elérhetőek.

kusan bővíthető. Erről bővebb útmutatóra a súgóoldalakon lelhetünk. A `$conn channel` mezőben lehet megadni azt a csatornát, amire a bot be fog lépni.
Most az eseménykezelő függvények következnének, de hogy érthetőbb legyen, mit miért tettünk, ugorjunk a forrás végére. Az eseményeket háromféleképpen kezelhetjük: az első esetben, ha nem határoztunk meg saját eseménykezelőt, az alapértelmezett hívódik meg, ami pusztán a legminimálisabb feladatokat látja el. A második esetben meghatározzuk, hogy egy esemény bekövetkeztekor melyik függvény hajtódjon végre (`add_global_handler()`). A harmadik esetben kapcsolatonként határozzuk meg, hogy az adott eseményre melyik függvény fusson le (`add_handler()`). Mivel csak egyetlen kapcsolatom van, bármelyik függvényt felhasználhattam volna a példában, de én mindkettőt alkalmaztam. Az `add_handler()` függvénynek meg kell adni az esemény azonosítóját és az eseménykezelő függvényt. Itt még nincs vége a variálhatóságnak, mivel utolsó értékeként megadható, hogy az alapértelmezett eseménykezelőt felülírja (0), illetve a sajátunk előtt (1) vagy után (2) futtassa le. Ha elhagyjuk ezt az értéket, akkor csak a saját eseménykezelő fut majd le. Hasonlóan kell meghívni az `add_global_handler()` függvényt is. Az események nevei a `Net::IRC::Event` modul forrásában találhatók meg. Ez több mint 160-féle eseményt különböztet meg. Én az 1. listán a következőket használtam:

- `msg`: privát üzenetet kap a bot.
- `public`: a csevegőszobába érkezik egy üzenet.
- `join`: valaki belép a csevegőszobába, még akkor is, ha maga a bot lép be.
- `part`: a csevegőszoba elhagyása.
- `376`: a MOTD végét jelző üzenet – ez egyértelműen azt mutatja, hogy a bot sikeresen csatlakozott.

Minden egyes eseménykezelő két értékkel hívódik meg: az egyik a kapcsolatot (`$conn`) írja le, a másik pedig az eseményt (`$event`). A `$conn->privmsg()` függvénnyel lehet üzenetet küldeni a csatornára, illetve egy adott személynek. Első értékében meg kell adni, hogy kinek (vagy melyik szobának) szól az üzenet, a második érték pedig az üzenet szövege maga. Az `$event` adatokat tartalmaz az eseményről: a csevegő nevét (`nick`), a gépének a nevét stb. A következőképpen lehet például egy csevegőnek opot adni:

```
$conn->mode( $conn->channel, "+o",
➥$event->nick );
```

Az 1. listában látható függvényeket nem részletezem tovább, ugyanis a megjegyzések alapján könnyen el lehet igazodni bennük.

További lehetőségek

Innentől kezdve a határ a csillagos ég: botunk már mindenre képes lehet, amit Perl nyelven meg lehet írni. Megkérdezhetjük az időt, a tévéműsort (➲ http://www.port.hu szűrése) és a napi horoszkópot, sőt botunk akár az összes csatornán elhangzott hivatkozást is megjegyezhetné. Ennek az utóbbinak a megvalósítását mutatja be a 2. lista, amelynek teljes forrása a CD-mellékleten található meg. A hivatkozásokat MySQL-adatbázisban tárolom, DBI segítségével (Debian: `apt-get install libdbi-perl`). Az egyszerűség kedvéért az adatbázis, a felhasználó neve és a jelszó is `timmy` lett. A hivatkozások táblát a következőképpen lehet létrehozni:

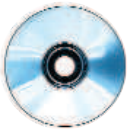
```
create table linkek( URL test, datum integer);
```

A MySQL használatának részletes bemutatása meghaladná e cikk kereteit. Aki még nem foglalkozott vele, az egyszerűen egy fájlban is tárolhatja a hivatkozásokat. Ha sikerült beállítani az adatbázist, akkor a botnak be lehet írni a csevegőszobába: ***http://www.linuxvilag.hu***, amit szépen tárol is. A lekérdezés a következőképpen néz ki: ***??URL [minta] [sorszám]***, itt mindkét érték elmaradhat, és mindig csak egy hivatkozást listáz ki. A sorszámozás pedig nullától indul és dátum szerint csökkenő sorrendben halad.
Ha eddig eljutottunk, és a hivatkozásokat már ragyogóan megjegyzi a bot, akkor egy kis átalakítással az is elérhető, hogy a csevegők aranyköpéseit is tároljuk és megőrizzük az utókornak. Aki nem szereti vagy nem ismeri a Perlt, annak a következő írásomat ajánlom, ami a mostanihoz hasonlóan saját IRC bot megírásáról fog szólni, de a Java nyelv használatával.

Kolcza Péter (kpeter@sysconfig.hu)
Imádja a South Parkot. A Miskolci Egyetem informatika szakos hallgatója. Elvakult GNU/Linux-rajongó. Ha egyetemi elfoglaltságai engedik, rendszerépítéssel foglalkozik. A cikkel kapcsolatban minden észrevételt szívesen fogad.

KAPCSOLÓDÓ CÍMEK

- ➲ http://www.irc.hu
- ➲ http://www.cpan.org
- ➲ http://www.mysql.com



Alkalmazzunk XML-t! (3. rész)

Ebben a részben szót ejtünk a program hiányosságairól, azután a Perl SQL-csatlakozás használatáról, végül pedig készülő weblapjaink képanyagának az elkészítéséről.

Elsőként – nem hagyhatom a végére, nehogy a kedvezőtlen benyomás maradjon meg – a hiányosságokról kell szólnom. Programunk felépítésének az a legfőbb hibája, hogyha egy tároló típusú elemet (amilyen például az előző részekben említett `lap1`) egy másik elem meghatározásában szeretnénk használni, az csak akkor működik az elképzeléseink szerint, azaz a végső hívásból véve a kimenet tartalmát, ha mi magunk (feltehetően hosszas gondolkodás után) megadjuk, hogy milyen szintről (veremszint) vegye a kimeneti adatot. A szintet a `level` kapcsolóval adhatjuk meg. Ez a hiányosság természetesen nagyon zavaró, de mivel csupán az elemek meghatározása során bukkan fel és a végső használatnál nem, úgy gondoltam, hogy ennek ellenére a program a gyakorlatban használható.
A másik nehézség abból adódik, hogy a program Perl-parancsfájlként működik, ennek megfelelően (különösen az első indításkor) elég lassú. Ezt hivatott kiküszöbölni a `-x` és `-o` kapcsoló, amelyek korlátozzák az elkészítendő fájlok számát. Ugyancsak gond, hogy bármilyen jó is az XML `MakeFile` a honlap oldalfelépítésének a leírására, nem biztosítja azt a rugalmasságot a célfájlok készítésére, amit az „igazi” `Makefile` és a `make` program használata (például a módosítások követése). Nos, ennyi talán legyen elég az önbecsmérlésből. Ki-ki keresheti a további hibákat, remélem, használat közben, nem pedig ex cathedra…

Az SQL-illesztés

Programunkhoz, tételezzük fel, létezik egy PostgreSQL adatbázisunk, valamint telepítve vannak a megfelelő programkönyvtárak (`Pg` Perl-modul). Ezután készítsük el a következő két táblát, az alábbiak szerint:

```
create table csoport (
       kod    text,
       nev    text );

create table arlista (
       csoport      text,
       nev          text,
       ar           float );
```

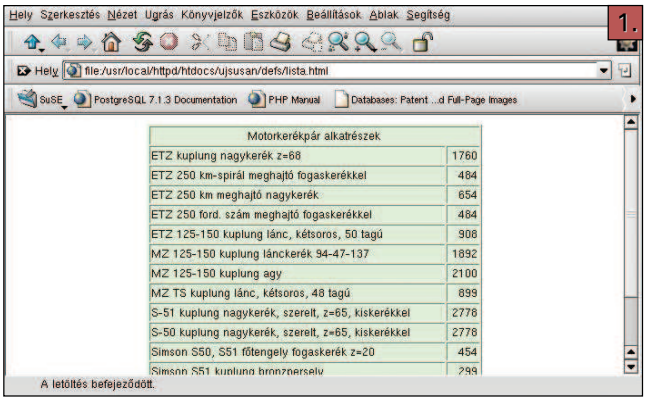
Vélhetőleg egy egyszerű, cikkcsoportokra osztott árujegyzéknek megfelelő ez a két tábla. Az `arlista` táblában a `csoport` mező idegen kulcs, a csoport tábla `kod` mezőjével áll kapcsolatban, ez viszont saját kulcs. Mindenesetre használhatjuk az általam mellékelt próbaadatokat. Figyelem, az árak elavultak!
Az SQL sajátosságainak megfelelően az illesztőfelület az alábbi módon réteges felépítésű:

- adatbázis;
- kérdés/eredmény (query/result);
- eredménysor (row);
- mezőérték (field);

A korábban már megfigyelt rugalmasság megvalósítása érdekében itt is igyekeztem minél szabadabbá tenni a használatot. Ezt hivatottak segíteni a különböző vermek. Ezek révén lehetséges több SQL-kérdés és eredménysor keverése, a név megadásával a megfelelő kiválasztása, illetve egyszerűbb esetben az alapértelmezett alkalmazása (a verem tetején lévő). Lássunk egy egyszerű példát! Ez a fent megadott adatbázisokból készít egy táblát olyan módon, hogy cikkcsoportonként külön kiírja az árukat. Ez idő tájt láthatjuk használat közben a korábbi részekben említett `_while` tagot is.

```
<html>
<table align='center' border='1'
bgcolor='#DDFFDD'>
<_sqldb name='susandb' dbname='susan'
➥user='avernus'>
<_sqlquery name='csoportok' query='select *
➥from csoport'/>
<_sqlrow name='csoport' qname='csoportok'/>
<_while test='sqltest("is_next","csoport")'>
<tr>
➥<td colspan='2' align='center' class='cimsor'>
<_sqlfn qname='csoportok' rname='csoport'
➥f='nev'/>
</td></tr>
<_sqlquery name='aruk' query='@sprintf("select
➥* from arlista where
➥csoport=&#39;%s&#39;",sqlfn("kod","csoportok",
➥"csoport"))'/>
<_sqlrow name='aru' qname='aruk'/>
<_while test='sqltest("is_next","aru")'>
<tr><td align='left'>
<_sqlfn qname='aruk' rname='aru' f='nev'/>
</td>
<td align='right'>
<_sqlfn qname='aruk' rname='aru' f='ar'/>
</td></tr>
<_sqlrow name='aru' qname='aruk'/>
</_while>
<_sqlrow name='csoport' qname='csoportok'/>
</_while>
</_sqldb>
</table>
</html>*
```

Egy adatbázis-kapcsolatot hozunk létre az `_sqldb` taggal. A `name` érték a későbbi hivatkozásra szolgál, így tetszőleges egyedi szó lehet. A másik két érték (`dbname` és `user`) a PostgreSQL-adatbázis neve és az engedéllyel bíró felhasználónév (nincs külön azonosítás). Mint látható, az `_sqldb` tartalmazza a többi SQL-es elemet, így nem kell megadni bennük az adatbázisnevet (a verem felső eleme).
Ezután az `_sqlquery` taggal felteszünk egy kérdést az adatbázisnak (`"select * from csoport"`). Az `_sqlrow`



Motorkerékpár alkatrészek	
ETZ kuplung nagykerék z=68	1760
ETZ 250 km-spirál meghajtó fogaskerékkel	484
ETZ 250 km meghajtó nagykerék	654
ETZ 250 ford. szám meghajtó fogaskerékkel	484
ETZ 125-150 kuplung lánc, kétsoros, 50 tagú	908
MZ 125-150 kuplung lánckerék 94-47-137	1892
MZ 125-150 kuplung agy	2100
MZ TS kuplung lánc, kétsoros, 48 tagú	899
S-51 kuplung nagykerék, szerelt, z=65, kiskerékkel	2778
S-50 kuplung nagykerék, szerelt, z=65, kiskerékkel	2778
Simson S50, S51 főtengely fogaskerék z=20	454
Simson S51 kuplung bronzperselv	299

taggal lehívjuk az első eredménysort. Ezután az `_while` tag segítségével végigmegyünk az eredménytáblán. Ebbe a `while` ciklusba ágyazva készítünk egy másik kérdést, amely a pillanatnyi csoport tagjait adja eredményül. Ehhez sajnos használnunk kell egy beépített függvényt, amely visszaadja egy adott kérdés–eredménysor–mező hármas értékét. Mivel ez a függvény nem ismeri az XML-elemzés helyzetét, így az SQL-vermekről sincs tudomása, sajnos mindhárom értéket meg kell adni. Újabb kényelmetlenség…
Az `_sqlfn` tagokkal a kimeneten elhelyezzük a megfelelő mezők értékét (meg van adva a kérdés, a sor és a mezőnév is, mivel jelenleg nem tartalmazzák ezeket az elemeket). Ha sikerül lefordítanunk ezt az állományt, akkor az *1. képen* láthatóhoz hasonló táblázatot kapunk eredményül.
Mint látható, a csoportokat külön sorban emeli ki, a mezők megfelelően vannak igazítva. Ennek a megoldásnak az igazi szépsége abban rejlik, hogyha az adatbázis módosítása után a fájlt újrafordítjuk, akkor mindennemű pluszmunka nélkül a pillanatnyi eredményt kapjuk meg. Így könnyen el lehet – például ismertető CD írásakor – a pillanatnyi listát készíteni, és nyomban fel is lehet írni, mindezt önműködően.

A képátalakító illesztése

Létezik egy nagyszerű képátalakító programkönyvtár, mégpedig az ImageMagick. Legfőbb előnye az, hogy mindenféle programnyelvhez megoldott az illesztése, így az átalakítások remekül gépesíthetők. Én magam is használtam a C, C++ és Perl nyelvi felületét. Ráadásul az illesztőfelületek felépítése hasonló (az objektumközpontú változatok hash értéklistát tudnak fogadni). Természetesen ne várjunk olyan mérvű hatásokat, mint amilyeneket a Gimppel érhetnénk el, viszont egyszerűbb forgatásra, átméretezésre, vágásra, beillesztésre, keretezésre, képformátumváltásra kitűnően használható, viszonylag gyors és kényelmes.
E rövid bevezető után vágjunk azonnal a közepébe! A legfontosabb, gyakran használt műveleteket a szokott módon: külső XML-modulok segítségével vittem be a programba. Lássuk a modulokat:

```
<convert file='bemenetifájl_neve'
target='kimenetifájl_neve' format='formátum'
quality='szám'/>
```

Mint nyilvánvaló, ez egy nem tároló típusú elem, azaz önmagában zárt. A be- és kimenet egy-egy képfájl, a formátum a jellegzetes kiterjesztéssel kerül megadásra, például jpg vagy png. A ***quality*** tulajdonság főleg jpg-nél szükséges, a minőséget adja meg, mivel a jpg veszteséges tömörítés, és a minőség határozza meg, hogy mennyire hű az eredetihez. Természetesen a jobb minőség nagyobb fájlméretet eredményez. A 75 százalék megfelelő szokott lenni.

```
<cropper file='bemenetifájl_neve'
grdir='képek_könyvtára'> ... </cropper>
```

Ez az elem a vágástároló eleme. Itt adjuk meg a vágandó fájlt, valamint az esetleges elérési útját. Ezután az alábbi `crop` elem segítségével tudjuk darabolni. Ez a felépítés azért jó, mert sokszor egy nagyobb képet több, esetleg sok részre kell vágni a grafikus felépítésű oldalakhoz.

```
<crop target='célfájl_neve' x='xpozíció'
y='ypozíció' w='szélesség' h='magasság'/>
```

Ez a tényleges vágóelem. A fenti `cropper` elemben lehet használni. Az ott megadott fájlból vágja ki az ***xy*** pozíció és a szélesség/magasság számnégyessel megadott részletet, és a megadott célfájlba rakja. A célfájl kiterjesztése adja meg a kimeneti formátumot.

```
<placer file='bemenetifájl_neve'
target='célfájl_neve'> ... </placer>
```

Ez a beillesztés tárolóeleme. A vágással ellentétben itt már meg van adva a célfájl, így egy (változatlanul maradó) kiinduló fájlba beszúrjuk az alábbi `place` elemmel megadott részleteket, majd a `target` értékben megadott néven mentjük (a formátumot itt is a kiterjesztés adja meg).

```
<place file='forrásfájl_neve' raise='hash'
scale='hash' border='hash' rot='fok'
x='xpozíció' y='ypozíció' />
```

Íme, a beillesztőelem. Elég bonyolult, mert a beszúrás előtt műveleteket is tud végezni a bemeneti fájlokon. A `rot` érték segítségével forgathatunk, fokban lehet megadni. Megjegyzem, hogy az ImageMagick bizonyos változatainál a forgatás eltolódással is jár, ha nem 90 vagy 180 fokról van szó. Ezt ki kell próbálni. A `scale` értékkel át lehet méretezni a beillesztendő képet. A megadandó indexelő nyalábolás (hash) formája:

```
'geometry => "800x600"'
```

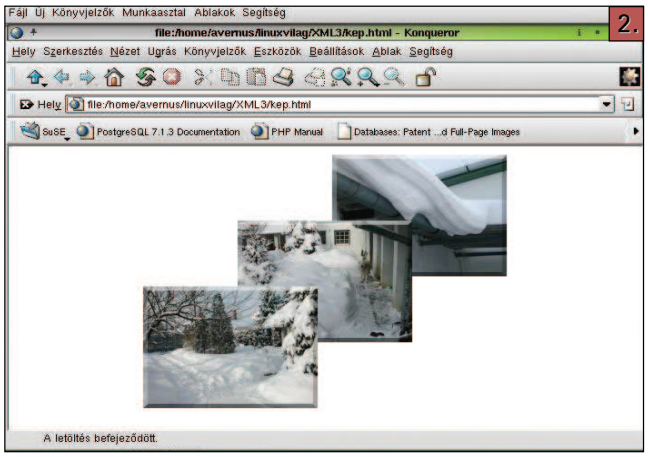
Természetesen a kívánt méreteket szükséges megadni.
A `raise` a kiemelést, illetve a besüllyesztést végzi, például

```
raise='geometry => "5x5", raise => "1"'
```

Az 1 azt jelenti, hogy kiemelésről van szó, a 0 pedig bemélyítést jelent.

```
<write x='xpozíció' y='ypozíció'
font='ttf_fontnév' size='fontméret'> ...
szöveg ... </write>
```

Ezzel az elemmel lehet a képre szöveget felírni. Ha akad TrueType betűkészletünk, azt megadva használhatjuk. A nyitó- és zárótag között kell lennie a szövegnek. Ha ez különleges karaktereket is tartalmaz, célszerű a `CDATA` határolót használni, hogy ne zavarja meg az XML-elemzőt. Sajnos jelenlegi, egyszerű formájában semmiféle tördelést nem végez a szövegen,



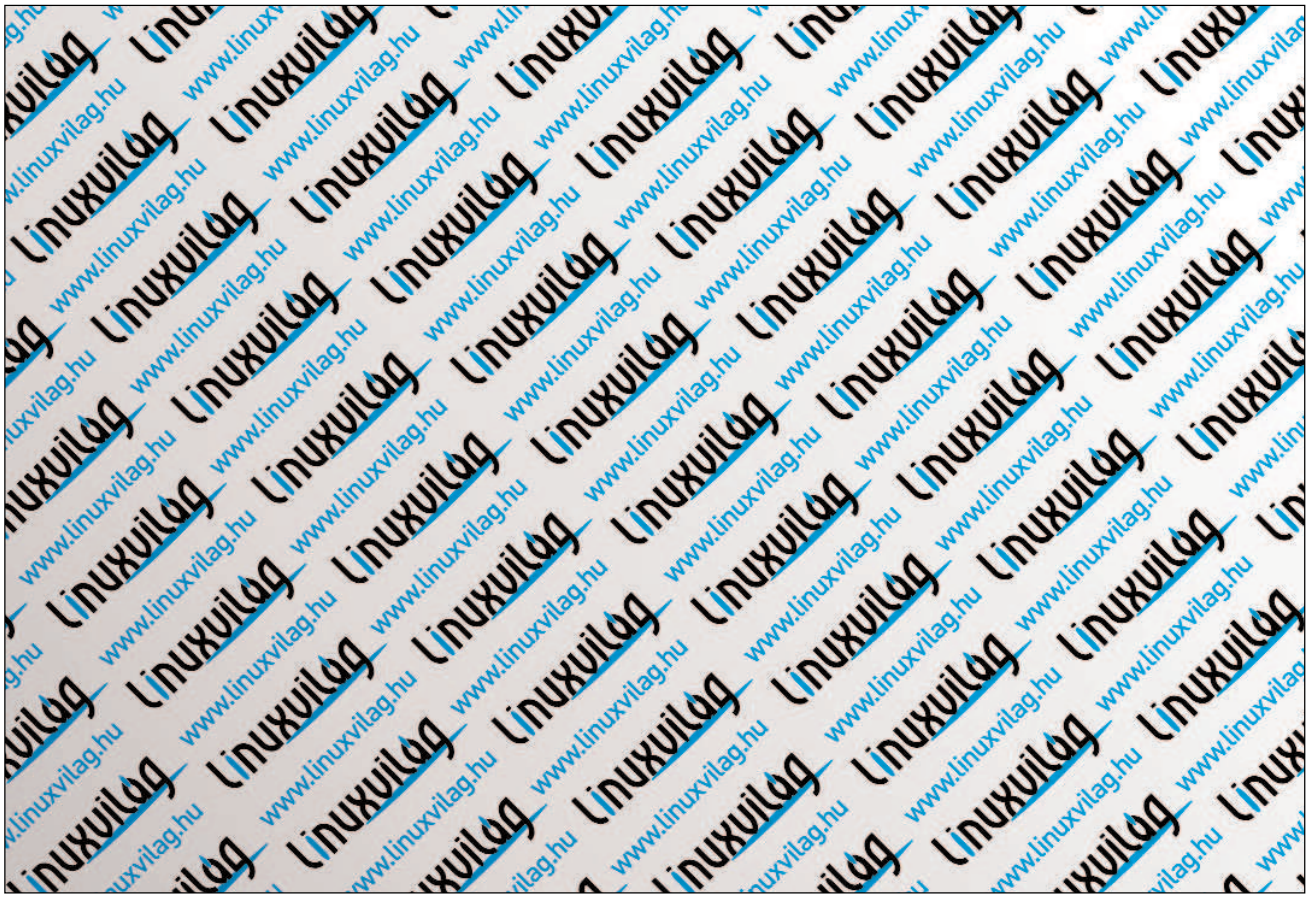
2.

azaz soronként kell bevinnünk, ügyelve, hogy ne lógjon le a képről (egy `write` elem/sor). Erre is lássunk egy egyszerű példát! Tegyük fel, hogy van három képünk, amelyeket lépcsőzetesen szeretnénk elhelyezni. Készítsünk a cél méretének megfelelő üres képet (például a Gimppel), legyen, mondjuk 400×300 pont. A háttért állítsuk átlátszóra. Nevezzük el ***ures.png***-nek!
Ezután használhatjuk a következő XML-fájlt a kívánt cél elérésére:

```
<_dummy>
<placer file='ures.png' target='cel.png'>
 <place file='ho1.jpg' x='208' y='0'
 scale='geometry => "192x144"'
 ➥raise='geometry => "5x5", raise => "1"'/>
 <place file='ho2.jpg' x='104' y='78'
 ➥scale='geometry => "192x144"'
 ➥raise='geometry => "5x5", raise => "1"'/>
 <place file='ho3.jpg' x='0' y='156'
 ➥scale='geometry => "192x144"'
 ➥raise='geometry => "5x5", raise => "1"'/>
</placer>
</_dummy>
```

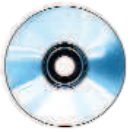
Ha ezt az `xtend fájlnév` paranccsal lefordítjuk, akkor a *2. képet* kapjuk eredményül (ebben az esetben egy HTML-fájlba helyezve).
Azt hiszem, az alapelv érthetővé vált. Maga az xml formátum csak adathordozóként szerepelt itt. De – és ezt fontos megérteni – másutt is csak erről van szó! Az XML nem csodaszer, önmagában nem több, mint egy jól meghatározott leíró formátum. De éppen azért, mert jól meghatározott és bővíthető, és mivel kitűnő elemző könyvtárak állnak hozzá rendelkezésre, nagyon jól használható különféle programjaink adattárolására és bemeneti állományának.
További ötleteket szívesen fogadok a hf@hmvhely.hu címen.

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



Az XML és a Perl (3. rész)

Húzd szorosabbra!



A legutóbbi részben ***olvas.pl*** néven közösen elkészítettünk egy nem túl összetett programot egy igen egyszerű XML jelölőnyelvvel írt dokumentum nyomtatható szöveggé alakításához. Nagy vonalakban így írtuk le a nyelvet: a dokumentum egy vagy több fejezetből áll, amelyek egymást követő bekezdéseket és beágyazott fejezeteket tartalmaznak. Minden fejezetnek van egy címe, és természetesen minden bekezdés karakterekből épül fel.
Bizonyára sokaknak feltűnt, hogy a program nem tiltakozik, ha a bemenet jól formázott XML ugyan, de nem felel meg a fenti leírásnak. Ha például egy olyan dokumentumot készítesz, amelyik a `<bekezdes>` elemen kívül is tartalmaz szöveget, az ***olvas.pl*** szó nélkül figyelmen kívül hagyná a karaktereket. Eddig azért nem beszéltem sokat az „alkalmazásszintű ellenőrzésről”, mert nem akartam, hogy a túl sok részlet között elvesszen a lényeg. Az alkalmazásszintű ellenőrzés azonban az XML-feldolgozásnak nagyon fontos része, így itt az ideje, hogy bővebben is bemutassam.

Szabályos nyelvtan

Említettem, hogy el lehet készíteni a jelölőnyelv nyelvtanát, az úgynevezett DTD-t (Document Type Definition), amely a megengedett elemeket egy XML leírónyelvben írja le. Egy ellenőrző értelmező ez alapján vizsgálhatja meg a dokumentumot. Bár mi nem használunk ellenőrző értelmezőt, egy DTD megírása így is hasznos gyakorlat lehet a nyelv szerkezetének leírásához.
Egy XML DTD-jelölőmeghatározásokból épül fel, amelyek `<!*` és `**>` között állnak. A jelölőmeghatározásoknak számos típusa létezik az XML-ben (és még több SGML-ben), de mi egyelőre csak az elemtípus-meghatározásokkal és a tulajdonságlista-meghatározásokkal foglalkozunk.
Egy elemtípus-meghatározás adja az elem nevét és a tartalommodellt, ez az elem lehetséges tartalmáról ad leírást. A tartalommodell valamennyire hasonlít a Perl szabályos kifejezéseihez. Készítsünk egy elemtípus-meghatározást `<dokumentum>` elemünkhöz:

```
<!ELEMENT dokumentum (fejezet+)>
```

Ez emberi nyelvre lefordítva mindössze annyit állít a `<dokumentum>` elemről, hogy egy vagy több `<fejezet>` elemet tartalmazhat. A `+` pontosan ugyanazt jelenti, amit a Perl a szabályos kifejezésekben tesz, így a `*` és a `?` is. Ha ezt írnánk:

```
<!ELEMENT dokumentum (fejezet)*>
```

azzal azt mondanánk, hogy a `<dokumentum>` nulla vagy több `<fejezet>`-ből állhat. A DTD elkészítésekor nagyon gondosan járjunk el, hogy a későbbiek folyamán ne kelljen felesleges ellenőrzőkódot írni a programunkba.
Most határozzuk meg a `<fejezet>` és `<bekezdes>` elemeinket:

```
<!ELEMENT fejezet (bekezdes | fejezet)*>
<!ELEMENT bekezdes (#PCDATA)>
```

A függőleges vonal a logikai „vagy” jele. A `#PCDATA` jelentése a „csak karakterek, semmi egyéb”, és valójában a „parsed character data” rövidítése, amelyben a „parsed” abban az értelemben áll, hogy már átfutott egy értelmezőn, és nem tartalmaz elemeket vagy hasonló XML-eszközt. Emiatt a `<`, `&` és a hasonló karaktereket a HTML-ből esetleg már megismert kóddal kell helyettesíteni, úgymint `&lt` és `&amp`.
A `<dokumentum>` és a `<fejezet>` tartalommodellje az, amit elemtartalomnak hívnak. Az ilyen tartalommal bíró elemek szöveget nem tartalmazhatnak, csak más elemeket.
A `<bekezdes>` tartalommodellje viszont úgynevezett vegyes tartalom – ez már szöveget is tartalmaz, és lehetséges volna olyan modellt készíteni, amiben elem és szöveg egyaránt megtalálható.
A szóköz, tabulátor, sortörés stb. kivétel képez az alól a szabály alól, hogy elemtartalomban nem szerepelhet szöveg. Másképp nem lehetne szépen megjeleníteni egy XML-dokumentumot, sok esetben még kézzel elkészíteni sem. Egy XML-értelmező nem hagyhatja figyelmen kívül az ilyen karaktereket, át kell őket adnia az alkalmazásnak, mintha vegyes tartalomban fordulnának elő. Ugyanakkor egy ellenőrző értelmező köteles szólni az alkalmazásnak, ha az említett karakterek az elemtartalomban jelennek meg; egy szövegformázó program valószínűleg figyelmen kívül hagyja őket.
Végül meg kell határoznunk a `<fejezet>` megengedett tulajdonságait (attributums):

```
<ATTLIST fejezet
       cim CDATA #IMPLIED >
```

Egy tulajdonságlista-meghatározás több tulajdonságot is felsorolhat, ebben az esetben azonban csak egyet tüntettünk fel, a ***cim***-et. A `CDATA` jelentése „character data”, ami egyszerű szöveget takar (ez viszont nem tartalmazhat közvetlenül például < nagyobb jelet). A `#IMPLIED` azt jelenti, hogy nem kötelező a tulajdonság megadása, és elhagyása esetén a feldolgozó alkalmazás alapértelmezett értékkel látja el. Ennek ellenére a tulajdonság nem állhat érték nélkül (lásd a Linuxvilág 32., októberi számának 70. oldalán). Ha ezen a helyen `#REQUIRED` állna, a tulajdonság elhagyása hibát okozna. Továbbá a `#IMPLIED` helyén egyszerű szöveg is szerepelhetne, amely hiányzó tulajdonság esetén alapértelmezett értékként viselkedne.
Az XML DTD sokkal több, mint amennyit itt össze tudtam foglalni. Nagyobb részletességgel szolgál az XML-ajánlás, és remek leírásokat találsz a weben, ha beizzítod a Google-t.

Kötelező nyelvtan

Ha az ***olvas.pl*** előző változatát kötelező nyelvtannal akarjuk felvértezni, ellenőriznie kell az elemek és a karakterek megfelelő helyen történő előfordulását. Jogos a kérdés, hogy miként tehetem ezt meg egy folyamalapú értelmezővel?

Igazság szerint az `XML::Parser` sokkal több, mint aminek az előző részekben elmondtam, és tartalmaz némi korlátozott lehetőséget az ellenőrzésre. Pontosabban fogalmazva a segítségével kitalálhatod, hogy a jelenlegi elem mely elemekbe lett beágyazva. Mint azt tudod, az első érték, amit az értelmező egy esemény kiváltásakor a meghívásra kerülő függvénynek átad, egy hivatkozás az értelmező objektumra. Ennek az objektumnak a típusa valójában az `XML::Parser::Expat`, és nem az `XML::Parser` maga. Az `XML::Parser::Expat` POD-jának gondos átolvasásával az ellenőrzés számos módszerét ismerheted meg. Az `in_element()` és a `within_element()` elemfüggvények különösen hasznosak a feladat megoldására. Álljon itt az ***olvas1.pl***, amely az ***olvas.pl*** hibaellenőrzéssel kibővített változata:

```
#!/usr/bin/perl -w
use strict;

use XML::Parser;
use Text::Wrap;
use Getopt::Std;

my ($indlevel,@sectnums,$parabuf,%opts);

getopts('c',\%opts);
die "Usage: ".$0." [-c] file\n"
➥unless @ARGV == 1;
my $p = new XML::Parser
➥(Style => 'Stream',ErrorContext=>2);
```

Az `ErrorContext` azt mondja meg az értelmezőnek, hogy hiba esetén hány sort mutasson meg a hibát tartalmazó sor előtt és után:

```
$p -> parsefile ($ARGV[0]);

sub StartTag {
     my ($expat,$eltype) = @_;
     if ($eltype eq "dokumentum") {
       $expat->xpcarp("<dokumentum> itt nem
       ➥fordulhat elo") if $expat->depth();
```

Az `xpcarp()` elemfüggvény segítségével egy figyelmeztető üzenetet jeleníthetünk meg a hiba környezetével együtt. A `depth()` elemfüggvény adja meg, hogy a jelenlegi elem hány elembe van beágyazva. A beágyazás számlálója a `StartTag()` meghívása után növekszik, így ez az ellenőrzés minden `<dokumentum>`-ot visszadob, a gyökérelemet leszámítva.

```
       $indlevel = -1;
       $sectnums[0] = 0;
       $parabuf = "";
       return;
     }
     $expat->xpcroak("<dokumentum> kell,
     ➥hogy a gyoker elem legyen") unless
     ➥$expat->within_element("dokumentum");
```

Ha az elemünk nincs egy `<dokumentum>`-on belül, feladjuk az értelmezést (ha nem tennénk, minden ezt követő elem esetén egy figyelmeztetést adnánk ki). Az `xpcroak()` hasonlít az `xpcarp()`-ra, de ez a figyelmeztetés mellett megszakítja a program futását. A `within_element()` nem figyeli a beágyazás mértékét. Akkor ad hamis értéket, ha a beágyazás egyik szintjén sem talált `<dokumentum>` elemet.

```
     if ($eltype eq "fejezet") {
       if (not
       ➥($expat->in_element("dokumentum")
       ➥or $expat->in_element("fejezet"))) {
           $expat->xpcarp("<fejezet> itt
           ➥nem fordulhat elo");
       }
```

Az `in_element()` azt nézi meg, hogy a jelenlegi elem melyik elembe van közvetlenül beágyazva. A fenti ellenőrzéssel megbizonyosodhatunk arról, hogy a `<fejezet>` csak a `<dokumentum>`-on vagy egy másik `<fejezet>`-en belül fordulhat elő.

```
       ++$sectnums[++$indlevel];
       $sectnums[$indlevel+1] = 0;
       print ' 'x(4*$indlevel),join
       ➥('.',@sectnums[0..$indlevel]),
       ➥" ",$_{cim},"\n";
       print "\n" unless $opts{c};
     } elsif ($eltype eq "bekezdes") {
       $expat->xpcarp("<bekezdes>
       ➥itt nem fordulhat elo") unless
       ➥$expat->in_element("fejezet");
     } else {
       die "Ismeretlen elem: ",$eltype,"\n";
     }
}

sub Text {
  my $expat=shift;
  tr/\n/ /;
  s/^\s+//;
  s/\s+$//;
  return if $_ eq "";
  $expat->xpcarp("szoveg itt nem fordulhat elo")
       ➥unless $expat->in_element("bekezdes");
```

Figyeld meg, hogy csak azután ellenőrzünk, miután megbizonyosodtunk róla, hogy a szöveg nem csak az angol szakzsargonban whitespace-nek nevezett karaktereket tartalmaz. Ha nem így tennénk, gyakorlatilag minden elem után figyelmeztetést kapnánk.

```
     $parabuf = $_;
}

sub EndTag {
     my ($expat,$eltype) = @_;
     if ($eltype eq "dokumentum") {
     } elsif ($eltype eq "fejezet") {
       --$indlevel;
     } elsif ($eltype eq "bekezdes") {
       my $ind = ' ' x (4*$indlevel);
       print wrap($ind,$ind,$parabuf),
       ➥"\n\n" unless $opts{c};
     }
}
```

A zárócímkéknél semmilyen ellenőrzésre nincs szükség, hiszen magának az értelmezőnek kell szólnia, ha ezek nem a megfelelő helyen állnak. Egy jellegzetes folyamalapú értelmezővel dolgozó alkalmazásnál igen gyakori, hogy az ellenőrzés és a változók használatra való felkészítése a nyitó címkék feldolgozásánál, míg az eredmény visszaadása a zárócímkéknél történik.

Faalapú értelmezés

Az előző hónapban több, kizárólag faalapú feldolgozásra tervezett modult is megemlítettem. Valójában az `XML::Parser` is nyújt két faalapú stílust. A „Tree" stílus egy nagyon egyszerű faszerkezetet hoz létre, melynek csomópontjai Perl-tömbökre mutató hivatkozások. Két típusú levél létezik:

- A szövegcsomópont: a tömb első tagja üres, míg a második maga a szöveg.
- Az elemcsomópont: a tömb első tagja az elem neve, a második pedig hivatkozás egy tömbre, amely az elem tartalmát foglalja magában. Az utóbbi tömb első tagja egy hivatkozás egy asszociatív tömbre (hash), amely az elem tulajdonságait sorolja fel az értékeikkel együtt. A további tagok szöveg- vagy elemcsomópontok.

Amikor a Tree stílust használod, az `XML::Parser parse()` és a `parsefile()` elemfüggvényei egy, a gyökérelem csomópontjára való hivatkozást adnak vissza.
Habár a Tree stílus leírása az `XML::Parser` utóbbi változataiban határozottan tisztább és áttekinthetőbb a réginél, ennek ellenére a használt adatszerkezeteket megérteni nem a legegyszerűbb dolog a világon. Ne legyen rossz érzésed, ha első nekifutásra nem értetted meg teljesen a fenti magyarázatot. Ha megnézted a következő példaprogramot, újra elolvastad a fentieket, ismét átnézted a kódot, előbb-utóbb rájössz, mit is jelent a „heuréka!" kifejezés. Az egyik ok, amiért érdemes lehet faalapú értelmezőt használni a folyamalapúval szemben, az az, hogy az értelmező egyszeri meghívásával ugyanazt a tartalmat többször akarod elemezni. Változtassuk meg az ***olvas.pl*** feladatát: ahelyett, hogy vagy a tartalomjegyzéket írja ki, vagy a tartalmat, inkább írja ki mindkettőt: előbb a tartalomjegyzéket, s ezt kövesse a tartalom. Ezt megtehetnénk egy folyamalapú értelmezővel is, ha két átmeneti tárban (buffer), külön tárolnánk a kimenetet, de az nem lenne olyan szórakoztató. Inkább felépítünk egy fát és kétszer bejárjuk. Az első bejárás alkalmával kiírjuk a tartalomjegyzéket és a második bejárás során a teljes tartalmat. Lássuk (ez ugyancsak megtalálható az 53. CD-melléklet Magazin/XML_Perl/olvas2.pl könyvtárában)!

```
#!/usr/bin/perl -w
use strict;

use XML::Parser;
use Text::Wrap;

my ($indlevel,@sectnums);

die "Usage: ".$0." file\n" unless @ARGV == 1;
my $parser = new XML::Parser
➥(Style => 'Tree',ErrorContext=>2);
my $tree = $parser->parsefile($ARGV[0]);
die "<dokumentum> nem a gyokerelem"
➥unless $tree->[0] eq "dokumentum";
```

Ha minden rendben van, a `$tree` egy hivatkozás egy kételemű tömbre, amelynek az első tagja „dokumentum" (a gyökérelemünk neve), a második pedig egy hivatkozás egy tömbre a `<dokumentum>` tartalmával.

```
$indlevel = -1;
$sectnums[0] = 0;
for my $pass (0..1) {
  my $p = $tree->[1];
  for (my $i = 1; $i < @$p; $i += 2) {
      if (not $p->[$i]) {
        warn "szoveg nem allhat itt"
        ➥unless $pass or $p->[$i+1]=~/^\s*$/;
        next;
      }
      warn "<".$p->[$i]."> nem allhat itt"
      ➥unless $pass or $p->[$i]
      ➥eq "fejezet";
      process_sect($p->[$i+1],$pass);
  }
}
```

Átugorjuk a `<dokumentum>` tulajdonságait tartalmazó asszociatív tömböt (hiszen elvileg nincsenek is neki tulajdonságai, az esetleges hamis tulajdonságokat pedig figyelmen kívül hagyjuk). A megmaradt tagokon párosával megyünk végig. Ha a dokumentum nyelvtanilag helyes volt, vagy üres szövegcsomópontokból, vagy `<fejezet>`-ekből tevődik össze. Az utóbbiak feldolgozását a `process_sect()` függvényre bízzuk, mivel egy `<fejezet>`-ben előfordulhat újabb `<fejezet>`, s ezt a legegyszerűbben rekurzívan oldhatjuk meg.

```
sub process_sect {
  my ($p,$pass) = @_;
  my ($href,$ind);
  if ($pass == 0) {
    ++$sectnums[++$indlevel];
    $sectnums[$indlevel+1] = 0;
    $href = join('.',@sectnums[0..$indlevel]);
    $ind = $indlevel;
    $p->[0]{_href} = $href;
    $p->[0]{_ind} = $ind;
    print ' ' x (4*$ind),$href,
    ➥" ".$p->[0]{cim}."\n";
```

Első körben kiszámoljuk a fejezetszámot és a bekezdés nagyságát, és a `<fejezet>` elem külön tulajdonságaként tároljuk. A tulajdonságok a tömb első tagja által mutatott asszociatív tömbben tárolódnak, ezzel a módszerrel dinamikusan bővítettük az értelmező által visszaadott fát, ugyanis csökkenti a létrehozandó és karbantartandó adatszerkezetek számát. Ezután a tartalomjegyzékhez kiírjuk a fejezet számát és a címét.

```
    } else {
      $href = $p->[0]{_href};
      $ind = $p->[0]{_ind};
      print ' ' x (4*$ind),$href,
      ➥" ".$p->[0]{cim}."\n\n";
    }
```

A második körben előkaparjuk a fejezet számára és a bekezdés nagyságára vonatkozólag az első körben elrejtett adatokat. Ezt használjuk fel a kiíratáshoz a teljes dokumentum megjelenítésekor is.

Bricolage-riasztások

Kövessük nyomon honlapunk eseményeit kulcseseményekhez rendelt riasztásokkal.

A múlt hónapban elkezdtük nézegetni ***David Wheeler*** Bricolage nevű, hatékony, `mod_perl`, `HTML::Mason` és PostgreSQL alapokra épülő, nyílt forrású tartalomkezelő rendszerét (CMS). A tartalomkezelő program viszonylag új típusú webalkalmazás, célja a méretesebb honlaprendszerek kezelési nehézségeinek a kiküszöbölése. A Bricolage megjelenését széles körben új nyílt forrású sikertörténetként ünnepelték, amely fényesen bizonyítja, hogy egy üzleti program nem feltétlenül rugalmasabb vagy hatékonyabb ingyenes társainál. Nem szabad elfelejtenünk, hogy bár a CMS olyan webalapú alkalmazás, amely dinamikusan létrehozott kiszolgálóoldali alapoktól függ, a CMS kimenete általában statikus oldal lesz. Ezért aztán annak ellenére, hogy a Bricolage viszonylag nagy web-, illetve adatbázisprogram, amelynek testreszabásához nem árt némi kiszolgálóoldali programozási ismeret, mégis olyan alkalmazás, amelyet nap mint nap nem programozók használnak. A Salon.com-tól kezdve a MacWorld On-line-ig rengeteg szervezet használja már most is a Bricolage-t. Ugyanakkor további weboldalak ezrei, a Cnettől a LinuxJournal.com-ig, tartalomkezelő megoldásokat használnak a Vignette rendszerétől (összetett, üzleti és drága) a PHPNuke-ig (egyszerű, nyílt forrású és ingyenes).

Ebben a hónapban az egyik kedvencemnek, a Bricolage-nak a riasztásképességét nézzük meg. Ez lehetővé teszi, hogy a felhasználó különböző eseményeket kövessen nyomon a Bricolage rendszerén. A riasztások nemcsak tájékoztatnak bennünket az események bekövetkeztéről, hanem egyúttal képet kaphatunk belőlük a teljes rendszer működéséről is – megtekinthetjük az objektumok listáját, feladatokat rendelhetünk ezekhez az objektumokhoz, valamint láthatjuk, hogy pontosan mikor hívódnak meg az egyes riasztások.

Riasztások

A Bricolage, csakúgy mint a legtöbb CMS-program, a cikkeket különféle állomásokon keresztül mozgatja. A Bricolage alatt ezeket az állomásokat asztaloknak (desks) nevezik, utalva ezzel a program újságírói gyökereire. A történetet ennek megfelelően a szerkesztőasztalon (edit desk) kezdik, majd átkerülnek a másolóasztalra (copy desk), onnan a jogi asztalra (legal desk), végül a kiadói asztalra (publish desk), ahonnan már a hálózatra jutnak. Maroknyi csapatok által vezetett honlapok esetében még könnyű nyilvántartani, hogy melyik cikk éppen hol található. De ha egyszer túlléptünk a kis szerkesztőgárdán vagy a csekély cikkszámon, igen nehézkessé kezd válni az egyes cikkek nyomon követése. Az egyik lehetőség, amivel ellenőrizhetjük a munkafolyamatot, ha a munkaasztalokat egyesével rendszeresen végignézzük, hogy lássuk, éppen milyen történetek vannak rajtuk, majd ennek megfelelően megtesszük a szükséges lépéseket. Ez a nézegetés azonban idővel elég fárasztó, ráadásul könnyen előfordulhat, hogy az összes cikk helyett nekünk csak a hírek kategóriát kellene átnéznünk, esetleg csak egy adott szerző műveit. Továbbá nem lenne rossz a munkafolyamat-eseményekről elektronikus levélben értesítést kapni.

A Bricolage rendelkezik ilyen lehetőségekkel, mégpedig igazi nyílt forrású stílusban, lenyűgözően testreszabható módon. A riasztások létrehozásához és módosításához válasszuk a képernyő jobb alsó sarkában található rendszermenü alól a ***Alert types*** menüt (riasztástípusok). Ez a menü a rendszergazda menüi közé tartozik, tehát csak a rendszergazdai jogosultságokkal rendelkező felhasználóknak jelenik meg. A kapott képernyőn – mint azt már a Bricolage karbantartó menüinél megszokhattuk – lehetővé teszi, hogy rákeressünk egy riasztás nevére, vagy kikeressük az összes, az adott betűvel kezdődő riasztást.

Riasztások létrehozása

Új riasztástípust a ***create new alert type*** hivatkozásra kattintva készíthetünk. Itt egy rövid HTML-űrlaphoz jutunk, ahol meg kell adnunk azt a Bricolage-objektumot, amelyikre a riasztást kérjük. Tehát például akkor szeretnénk riasztást kapni, ha valami, a történeteket (story) érintő dolog történik, ekkor kérjük a ***Story*** objektum riasztását. Ugyanígy, ha új felhasználó felvételekor szeretnénk riasztást, kérjük a ***Users*** objektumhoz tartozó riasztását. Tömören, szinte minden Bricolage-objektumhoz rendelhetünk riasztásokat.

Példaképpen készítsünk egy olyan riasztást, amelyik akkor figyelmeztet, amikor valamely, a címében a „Linux” szöveget tartalmazó cikk az egyik asztalról a másikra kerül át. Ne feledjük, a Bricolage segítségével szinte minden objektum megfigyelhető. A következő példa kétségtelenül adja magát.

Válasszuk a ***create new alert type*** hivatkozást az ***Admin*** menüből, és készítsünk új riasztást a ***Story*** objektumhoz. Itt a Bricolage által megfigyelhető eseménylistához jutunk, a kategóriahozzáadástól kezdve a történetünk kiadásán át a történet valamely részének törléséig. Példánkban mi a ***Story moved to desk*** (a történet mozgatása asztalok között) lehetőséget választjuk, majd a ***Linux story moved*** nevet adjuk neki.

Minden riasztástípushoz tartozik egy tulajdonos; jelen esetben a tulajdonosok mi leszünk, hiszen a saját nevünkön léptünk be. A Bricolage egyetlen felhasználót készít el magától, a ***Bricolage Administrator***-t, mégpedig `admin` bejelentkezési néven. Az `admin` felhasználó a Unix-rendszerek `root` felhasználójának, vagyis a rendszergazdának felel meg. Nyilvánvaló módon riasztásokat is birtokolhat, de sokkal jobban járunk, ha készítünk magunknak egy másik felhasználót (a bal alsó sarokban található ***Admin/User*** menüpontban), majd megadjuk a rendszergazdai jogosultságokat és a továbbiakban így jelentkezünk be (a `root` helyett).

Valamely esemény bekövetkeztekor ennek a lapnak az alján található ***next*** gombra kattintva a fő riasztástípus-szerkesztő oldalra jutunk, ahol új riasztástípusokat hozhatunk létre, illetve módosíthatjuk a már meglévőket. Minden riasztás négy részből áll:

- ***Properties*** (Tulajdonságok): a riasztástípus neve és tulajdonosa, amit az előző oldalon gépeltünk be.

- ***Rules*** (Szabályok): megadja, hogy mikor kell a riasztásnak megszólalnia. Minden szabály tartalmaz egy változót (amit lenyíló menüből választunk ki, így elkerülhetjük az esetleges elgépeléseket), összehasonlító tesztet és egy szövegmezőt, ahová az összehasonlítandó értéket írhatjuk be.
- ***Content*** (Tartalom): az elektronikus levélben levő üzenet, amit a riasztás címzettjei megkapnak. Ebben több különféle változót is használhatunk, a cikk címétől kezdve a kiadás dátumáig.
- ***Recipients*** (Címzettek): azok a felhasználók és csoportok, akiknek a riasztást szánjuk. Tudunk például riasztást küldeni az összes szerkesztőnek, az összes szerzőnek vagy éppen Gyurinak és Ferinek, de Debinek és Máriának nem.

Riasztási szabályok

A Bricolage riasztási rendszerének talán legérdekesebb része a szabályok alkotása, hiszen olyan egyszerűen alkalmazhatjuk őket, hogy a programozáshoz semmit sem értők is elboldogulhatnak velük. Ugyanakkor a `=~` és `!~` műveleti jeleknek (amelyeket látva egy Perl-programozó rögtön szabályos kifejezésekre gondol) akad egy nagy veszélyük. Egyértelműen kétélű fegyverről van szó, hiszen a szabályos kifejezések rendkívül hatékonyak a felvilágosultak kezében, de hihetetlenül veszélyesek (és elkedvetlenítők) a tudatlanokra nézve.
Riasztásunk létrehozásához a bal oldali tulajdonságlistából kiválasztjuk a ***Story title*** elemet, majd a `=~` jelet az összehasonlító műveleti jelek (operator) listájából, végül a `Linux` szöveget gépeljük be a szövegmezőbe értékként. Így a következőt kapjuk:

```
Story title =~ Linux
```

A `=~` műveleti jelet és nem az egyszerű `=` kell használnunk, mivel a Linux szót a cím bármely részében keressük és nem csak a teljes címmel való egyezést fogadjuk el. Amennyiben a Linux és a Perl is érdekel minket, a következőre is kereshetnénk:

```
Story title =~ Linux|Perl
```

A tapasztalt Perl-programozók biztosan örömmel hallják, hogy a `=~` és a `!~` műveleti jelek itt nem figyelik a kis- és nagybetű közti különbségeket.
A riasztás témája és szövege tetszőleges szöveget tartalmazhat, ideértve Bricolage által előre meghatározott behelyettesíthető Perl-változókat is. Egy szép JavaScript-megoldással a Bricolage lehetővé teszi, hogy a változókat lenyíló menüből válasszuk ki, elkerülve ezzel a gépelési és egyéb hibákat. Egyben igen hasznos jegyzék, hiszen nem kell megjegyeznünk az összes elérhető, a cikkekhez tartozó változónevet. Ezzel a módszerrel könnyen összeállíthatjuk a következő alakú riasztáscímet:

```
A $title cikk átkerült a $desk asztalra
```

Bár a riasztásüzenet valószínűleg más szöveget tartalmaz, itt is használhatjuk a változóneveket. A szövegtestbe beírhatjuk például a következőket:
Ön kérte, hogy figyelmeztessük, amennyiben a Linux témájú cikkek új asztalra kerülnek át. Nos, $trig_full_name épp most helyezte át a "$title" című cikket a $desk asztalra. Reméljük, segítettünk!

A riasztások fogadása

Amikor egy riasztás életbe lép, a kiválasztott címzettek két különböző módon kaphatják meg az üzenetet. Valamennyi bejegyzett felhasználó levélben üzenetet fog kapni, amely a fent bemutatott levélben tájékoztatja őket a változás bekövetkeztéről. A Bricolage ugyanakkor fel is jegyzi ezeket az eseményeket a saját adatbázisában, ami igen megkönnyíti a riasztások webes követését. Amikor a fent leírt linuxos riasztás életbe lép, levélben jelzést kapunk. Ugyanakkor valamennyi riasztást megtekinthetjük a riasztási oldalunkon. Ezt a képernyő tetején található, a ***my alerts*** gombra kattintva tekinthetjük meg.
A riasztások egészen addig maradnak ezen az oldalon, amíg nem jelezzük, hogy tudomásul vettük őket. A riasztásoldalt nem hosszútávú tárolóhelynek szánták; sokkal inkább egyszerű üzenetküldő rendszernek, amely lehetővé teszi a szerkesztőknek, hogy egyszerre lássák a honlapon akkortájt történt összes lényeges eseményt.
A riasztásokat olyan módon vehetjük tudomásul (ezzel törölve őket az oldalról), ha a listában egy vagy több jelölőnégyzetet bekapcsolunk, majd a lista alján lenyomjuk az ***acknowledge checked*** gombot. A jelölőnégyzetek használata nélkül is tudomásul vehetjük az összes üzenetet egyszerre, ha az ***acknowledge all*** gombot választjuk.

Tanulságos riasztások

A riasztások nagyszerű és hasznos eszközök, amelyekkel a Bricolage képes a vonatkozó adatokat eljuttatni a megfelelő felhasználókhoz, és így nem kényszeríti rá őket, hogy maguk keresgéljék ki őket. Mindeközben áldásos mellékhatásként a friss Bricolage-rendszergazdáknak és -programozóknak segít megérteni, miképpen jönnek létre különféle események az egyes objektumokban. Egyszerű példa erre a ***User*** (felhasználó) objektum: készíthetünk olyan riasztást, amely akkor figyelmeztet, ha egy felhasználót létrehozunk vagy törlünk, illetve megváltoznak a beállításai vagy a jelszava. Az ilyesfajta riasztás egy szerkesztő szemében nem ér különösebben sokat, ugyanakkor elsődleges fontosságú lehet a rendszergazda számára. Valamivel összetettebb példa a ***Template*** (sablon) objektum, amellyel a következő hónapban ismerkedhetünk meg. A sablonok határozzák meg, hogyan jelenjenek meg a történetek, ezért igen fontos a megfigyelésük és a változásaik nyomon követése. Ezért aztán a sablonokhoz több beállítás tartozik, mint a felhasználókhoz – nyomon követhetjük a sablon telepítését, szerkesztését, akár azt is, hogy mikor kerül fel az asztalára. Amennyiben esetleg korábban nem értettük volna meg, hogy akárcsak a történeteknek, a sablonoknak is van saját asztaluk, a riasztórendszerből ez világosan kiderül.
Én elég sokat tudtam meg a Bricolage rendszerről, miközben különböző szemszögekből megvizsgáltam, ideértve a riasztásmegadásokat is. Ha még újak vagyunk a Bricolage rendszerben, és a dokumentumok végigolvasása és a menük kinyitogatása után (ami igen fontos, amennyiben tisztában szeretnénk lenni a Bricolage oldalszerkezetével és jogosultságrendszerével) sem igazán értettük meg a dolgok belső lelkivilágát, a riasztásmeghatározások végignézése rengeteg dologra rávilágít.
A cikkhez tartozó kapcsolódó címek megtalálhatóak az 53. CD Magazin/Bricolage könyvtárában.

Linux Journal 2003. október, 114. 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'inben él.

Tükröm, tükröm...

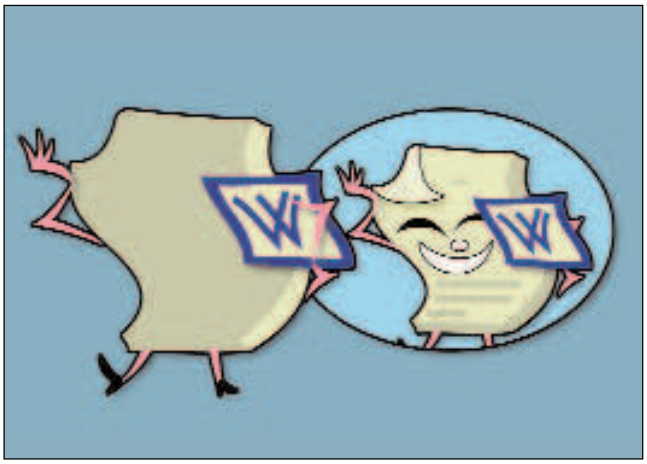
Készítsünk egyszerű biztonsági mentéseket, és tartsuk naprakészen az összes web- és FTP-oldalunkat olyan eszközökkel, amelyek már alighanem a rendszerünkön vannak.

François, mit művelsz? Amikor arra kértelek, hogy tükrözd a weboldalainkat, azt nem úgy értettem, hogy a képernyő elé kell tartanod egy tükröt. Néha nagyon bután tudsz viselkedni! Azt szeretném, ha arra a másik gépre másolatot készítenél a honlapunkról. François, most meg mit nézel annyira? Á, megérkeztek a vendégeink! Miért nem szóltál? Isten hozott titeket Chez Marcelnél, a linuxos fogások és a különleges borok otthonában.
Ha már a bornál tartunk, François, siess gyorsan a pincébe! Hozd fel, kérlek az 1999-es California Stag's Leap District Cabernet Sauvignont. Ez a merész és zamatos bor mai menünk tökéletes tükre. Mint már tudjátok, mes amis, mostani írásom témája a rendszerfelügyelet. A mai menü révén belekóstolunk számos, az adattükrözés céljára szolgáló eszköz működésébe. Rengeteg okot találhatunk az adatok tükrözésére: az első nyilvánvaló ok a biztonsági másolat készítésének talán nem túl vonzó, mégis nagyon fontos kérdése. Az egyéb okok közt találhatjuk meg a helyi hálózati frissítések számára létrehozott FTP-tükrözéseket, amilyen a saját RPM frissítőcsomagok tárhelyének létrehozása, vagy a weboldalak tükrözése a gyors, hálózati kapcsolat nélküli böngészés céljából.
Sokan azok közül, akik rendszeres időközönként biztonsági másolatokat készítenek, az adatokat egy másik gépük merevlemezén helyezik el, míg mások ugyanannak a gépnek egy másik lemezére dolgoznak. Mivel egy újabb merevlemez vásárlása manapság már rendkívül olcsó, és a nagy tárolókapacitású szalagos egységek lényegesen többe kerülhetnek, ez a megoldás is elterjedt.
Egy lemezről biztonsági másolatot készíteni egy másik lemezre olyan egyszerű is lehet, mint egy rekurzív másolás a `cp` paranccsal. Ha például a */**home*** könyvtáram minden adatát egy sok szabad hellyel bíró másik lemezre szeretném másolni, az alábbi parancsot használhatnám:

```
cp -rfupv /home/mgagne /disk2/
```

Ahogy azt már sejtitek is, a `-r` kapcsoló utal a rekurzív másolásra (az összes alkönyvtár másolására), a `-v` pedig a parancs végrehajtása során keletkező üzenetek kiírásáért felelős. Mivel nem akarom, hogy a rendszer minden egyes fájl felülírásakor figyelmeztessen, a `-f` kapcsolóval hagyom jóvá az összes másolást, a `-p` kapcsoló pedig azt biztosítja, hogy az engedélyek másolása is megfelelően megtörténik. Végül a `-u` kapcsoló arra utasítja a `cp` parancsot, hogy csak a megváltozott fájlokat másolja, ezáltal felgyorsítható a másolás folyamata.
Az eljárás viszonylag jól működik, de az egyik gépről a másikra történő másoláshoz további lépések is szükségesek. Linuxrendszerünkben a `cp` parancson túl számos eszköz áll még rendelkezésünkre. A kezdők számára egy teljes weboldal másolására az eredetileg ***Hrvoje Niksic*** által írt `wget` parancs kipróbálása az egyik lehetséges út:

```
wget -m http://www.websitename.dom
```



A `wget` a weboldal tetejétől kezdve a teljes oldalon végighalad, és mentést készít az ott lévő HTML-fájlokról és képekről. A `-m` ebben az esetben a tükrözést jelenti, de sok egyéb lehetőség is rendelkezésünkre áll a `-r`, `-N`, `-l inf` és `-nr` kapcsolók formájában. Ezek sorrendben arra utasítják a `wget`-et, hogy rekurzív letöltést végezzen, kapcsolja be az időbélyegzést, korlátlan számú szintet tegyen lehetővé, és ne távolítsa el az FTP-könyvtárak ***.listing*** fájljait.
A weboldal minden fájljáról másolat készül egy helyi könyvtárba, amelynek neve az oldal nevével egyezik meg. A fenti példában ez ***www.websitename.dom*** lenne. Ha a webkiszolgálónkon egy új fájlt hozunk létre és újra lefuttatjuk a parancsot, akkor csak ezt az új fájlt tölti le, így a másolat naprakészen tartása sokkal gyorsabb.
Ez az eszköz nagyszerűen használható arra a célra, amire írták, ez pedig elsődlegesen a weboldalakkal kapcsolatos, de a `wget` FTP-kiszolgálókról való adatletöltésre is lehetőséget nyújt. Ha névtelenül látogatható oldalról szeretnénk letölteni valamit, a parancsformátum szinte megegyezik a weboldal tükrözésére használt parancséval:

```
wget -m ftp://ftp.ftpsitename.dom
```

Ellenben ha olyan oldalról készítjük a biztonsági másolatot, amelynek az eléréséhez felhasználói név és jelszó is szükséges, egy kicsit cifrább a parancs formája:

```
wget -m
➥ftp://username:password@ftp.sitename.dom
```

Ennek a megközelítésnek van némi árnyoldala. Először is a jelszavunkat kódolatlan szövegként küldjük át a hálózaton, ami a hálózatunk megbízhatóságától függően elég kockázatos lehet. Szorult helyzetben az `scp` paranccsal készíthetünk biztonságos rekurzív másolatot. Mivel az `scp` az OpenSSH részét képezi, a parancs használatával élvezhetjük a

nyugalmat, hogy a fájlokat biztonságos, kódolt formátumban továbbítjuk. Feltéve, hogy – az Apache kiszolgáló gyökerétől kezdve – az egész weboldalt le szeretnénk másolni, a parancs valahogy így festene:

```
scp -rpv /var/www
➥root@remote_host:/mnt/backupdir
```

A `-r` kapcsoló felelős a rekurzív másolásért, a `-p` pedig arra utasítja az `scp`-t, hogy őrizze meg az eredeti fájlok és könyvtárak módosításának az idejét, a tulajdonosi és jogosultsági beállításokat. Amennyiben nagy mennyiségű adat átviteléről van szó, érdemes megfontolni a `-C` kapcsoló használatát, amely röptömörítést hajt végre, s ezzel lényegesen növeli az áteresztőképességet.
Az adattükrözés e módszereivel talán az a legnagyobb baj, hogy nagyon sokáig tartanak. A `wget` képes arra, hogy az új fájlokat egy FTP-kiszolgálóról letörölje, de arra már nincs lehetőség, hogy egy könyvtárat a fájlok esetleges törlésével teljes összhangban tarthassunk. Az `scp` is hasznos biztonságos másolása miatt, de arra nincs megoldás a programban, hogy csak a megváltozott fájlok kerüljenek átvitelre. Ez az eddigi megoldások második hátulütője. Ha biztosak szeretnénk lenni abban, hogy az adataink folyamatosan összhangban lesznek anélkül, hogy minden egyes fájlt és könyvtárat átmásolnánk, egy kicsit okosabb programra lesz szükségünk.
Ennek megoldására az általam ismert talán legjobb program ***Andrew Tridgell*** `rsync` nevű alkotása. ***Mick Bauer*** nagyszerűen bemutatta a programcsomagot (a Linuxvilág májusi és júniusi számában), úgyhogy én most nem szeretném az ott leírtakat ismételni, csak annyit mondanék, hogy esetleg érdemes ezt a kétrészes cikket újra fellapozni.

Az ftpcopy

Sok esetben öreg barátunk, az `ftp` társaságában találhatjuk magunkat. Az egyik oldalon (azon a gépen, amelyiket tükrözni akarjuk) az FTP-kiszolgálónkat futtatnánk, vagy a ProFTPD-t, vagy a `wu-ftpd`-t. A másik oldalon pedig ***Uwe Ohse*** `ftpcopy` programja futna. Az `ftpcopy` gyors, könnyen beállítható és egyszerűen használható program, amely igen jól teljesít, amikor teljes könyvtárszerkezetek másolása a feladat. A másolás folyamán gondoskodik a jogosultságok és a módosítás dátumának és időpontjának a kezeléséről, ráadásul mindezt gyorsan teszi. Mindezeken túlmenően nyomon követi a korábban letöltött fájlokat, ami célszerű, ugyanis a legközelebbi futtatáskor csak az időközben megváltozott fájlokkal kell foglalkoznia, így téve még gyorsabbá a biztonsági másolat készítését.
Néhány rendszercsomag már tartalmazza az `ftpcopy`-t, de a legfrissebb csomag letöltéséhez érdemes meglátogatnunk a ➲ http://www.ohse.de/uwe/ftpcopy/ftpcopy.html oldalt.
A csomag fordítása egyszerű, mindössze néhány lépésből álló feladat:

```
tar -xzvf ftpcopy-0.6.2.tar.gz
cd web/ftpcopy-0.6.2
make
```

A ***command*** nevű könyvtárban három futtatható állományt fogunk találni: az `ftpcopy`-t, az `ftpcp`-t és az `ftpls`-t. Ezeket akár innen is futtathatjuk, vagy átmásolhatjuk a ***/usr/local/bin*** könyvtárba, netán az `$PATH` útvonal valamelyik másik helyére. Nézzük, hogyan működik a program. Tegyük fel, hogy tükrözni vagy menteni szeretném a saját könyvtáramat egy távoli számítógépre. Az alap `ftpcopy` parancs valahogy így néz ki:

```
ftpcopy -u marcel -p secr3t! remote.hostname
/home/marcel /mirdir/
```

A `-u` és `-p` kapcsolók nyilvánvalóan a felhasználói nevemet és a (hamis) jelszavamat jelentik a távoli gépen. Ezt követi a másolni kívánt könyvtárra mutató elérési útvonal, ezután az a helyi könyvtár, ahol a könyvtárszerkezetet létre szeretném hozni. Az áttöltés folyamán ilyen üzeneteket fog kapni:

```
/mirdir/scripts/backup.log: download
                            ➥successful
/mirdir/scripts/checkhosts.pl: download
                               ➥successful
/mirdir/scripts/ftplogin.msg: download
                              ➥successful
/mirdir/scripts/gettime.pl: download
                            ➥successful
```

Ha ennél egy kicsit részletesebb adatokra van szükségünk, használhatjuk a `-bps` kapcsolót, amely az adatátvitel sebességét jeleníti meg bájt per másodperc mértékegységben. Érdemes legalább egyszer a `-help` kapcsolóval is lefuttatni az `ftpcopy`-t, és óvatosnak kell lennünk bizonyos lehetőségek használatával. A `-s` például a közvetett hivatkozásokra vonatkozik, a `-l` pedig a naplózási szint növelését teszi lehetővé. Ha a tükrözést egy `cron` munkafolyamatként szeretnénk beállítani, talán érdemes a naplózást `0`-ra állítani. Egy másik hasznos kapcsoló a `-n`. Ha egy fájlt töröltünk a távoli gépen, az `ftpcopy` futtatásakor helyileg is törlődni fog. Ha a rendszereket tényleg összhangban szeretnénk tartani, akkor erre van szükségünk. Ezt a műveletet kapcsolhatjuk ki a `-n` kapcsolóval, ebben az esetben nem következik be a törlés.
Nos, mes amis, elérkezett az idő, amikor mindnyájunknak haza kell térnünk. Azért egy utolsó pohár borra még futja az időből, non? François, mon ami, ha volnál szíves a házigazda szerepét magadra vállalni! És mindjárt két pohárral, hogy tükrözni tudjuk az elsőt!
Egészségetekre, mes amis! A vôtre santé! Bon appétit!

Linux Journal 2003. október, 114. szám

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

KAPCSOLÓDÓ CÍMEK

OpenSSH ➲ www.openssh.org
rsync ➲ rsync.samba.org
wget ➲ wget.sunsite.dk
ftpcopy ➲ www.ohse.de/uwe/ftpcopy/ftpcopy.html
Marcel borlapja ➲ www.marcelgagne.com/wine.html

Shogo: mobil harci hadosztály

Egy kis lövöldözés mindenkinek kell!

A Shogo: mobil harci hadosztály egyszemélyes lövöldözős játék, némi eltéréssel a többi hasonszőrű játékhoz képest. A programot olyan rajzelemek szövik át, olyan hatások nyoma figyelhető meg rajta, amelyeket a klasszikus japán rajzfilmekben, animációkban láthatunk. A fejlesztők a Macross (aka Robotech) és a Gundam grafikai eljárásokat használták, amelyek lehetővé teszik, hogy harci ütközetet vívj egy tíz méter magas robotban.
A játék időszerű, hihető történettel próbál meg végigvezetni bennünket az események folyamán.
A Shogóban Sanjuro alakjával játszhatsz, aki befolyásos parancsnokként a háború közepén elfogja az ellenséges terrorista fedőszerveket. Ahogy a fejlesztők kialakították Sanjuro világát, úgy fogod te is megismerni mindezt. A Shogóban megvan minden, aminek egy ádáz 3D-s akcióban lennie kell.
A játék néhány részében előfordul, hogy gyalogos katona vagy, míg másutt egy teljes, az életedet védő MCA egység, ami nem más, mint egy tíz méter magas fémruha. Ez a „csomag" valóban csodálatos tűzerővel rendelkezik, ami egy lézerágyút és egy multifunkciós harcifegyver-állványt foglal magában. A valós animált világban az MCA-k elképesztő mértékben irányíthatóak, és furcsa érzést kelt, amint összelapítják a gyalogos katonákat – emlékezz erre, és fuss, ha te is gyalogos katonaként küzdesz, és egy terrorista MCA jelenik meg…
A Shogo a Monolith LithTech 3D-motorját használja, amely a Quake II grafikai motorját kiegészítve képezi le a 3D-megjelenítést. Említsük is meg az eltéréseket, bővítményeket: a robbanások, a sebesülések, a szemcse- és a fényhatások olyanok, mintha valósak-valódiak lennének (ne feledjük, ez csak a Quake II szinthez képest előretörés). A Litgh-Tech-motor lehetővé teszi, hogy az MCA-ba bújva valóban meggyőző belső és külső látványt élvezhessünk, miközben ellenséges tankok vagy katonák támadnak.
A Shogo hangsávjai finomak és hatásosan illeszkednek a képernyőn megjelenő akcióra. Néhány hanghatás szabályosan nyomást gyakorol az animált történetre, és ez elborzasztóan meggyőző. Ezenkívül a Shogo jó minőségű hanganyaggal (narrátor) bír, ami a játék egészét végigköveti és irányítja a karakter életét.

A játék jellegzetességei dióhéjban

1. A LithTech 3D-motorját használja, amelyet a Monolith készített (➲ http://www.lith.com/lithtech).
2. Kiválasztható a pilóta, és egy a négy átalakítható Mobile Combat Armor eszköz közül (MCA-k).
3. Két módban lehet játszani: saját lábon állva és az MCA-ba bújva.
4. Igazi 3D-környezet fogad bennünket valós fényekkel, belső szintekkel, hatalmas külső tereppel és félelmetes hatásokkal.
5. Több mint harmincféle kihívó ellenség, akik ügyesen kiaknázzák a környezet nyújtotta lehetőségeket.
6. Mozgalmas történet és célközpontú küldetési szerkezet, ez határozza meg a támadás alapjait.
7. Elkülönített MCA- és gyalogos fegyverarzenál (18 felturbózott változat).
8. Többjátékos akció, ami tartalmazza a beépített internet-, helyi hálózat- és modemtámogatást.

- ***LithTech-jellemzők: 3D-modellezés***
  - Támogatja a mozgásalapú animációt.
  - Részletes megjelenítés a külső világokhoz.
  - Egyéni textúrák a modellekhez.
  - Többszintű modellek: mindegyik modell százszintes, kidolgozott megjelenéssel bír – mindehhez nem szükséges többletmemória, a modellek lassúbb rendszerekre is visszaalakíthatók.
  - Az összes modell hierarchikus, a kulcsfontosságú animációs részek interpoláltak.
  - Támogatja mind a merev, mind a deformált animációs megjelenítést, lehetővé teszi az élvezetesebb és jobb minőségű életszerű animációt.
- ***LithTech-jellemzők: fények és különleges hatások***
  - csillogások, színes fények, dinamikus fénykezelés;
  - átlátszóság és visszaverődés a víz- és üveghatásokhoz;
  - valós idejű árnyékok;
  - tájak, városok;
  - animált felhők, színes köd;
  - nagymértékben testreszabható szemcsehatásrendszer;
  - PolyGrids (elmozdítható, változó grafikai elemek például a vízfelszín létrehozásához).

Ismerkedjünk meg a szereplőkkel!

Sanjuro Makabe

Sanjuro Makabe fiatal parancsnok az UCA védelmi erőknél. Láthatod, amint egyre tiszteletreméltóbbá válik, te irányítod az útját a teljes elismertségig. Így te vagy az is, aki később, a küldetés során felelős leszel egy tragédiáért, ami társaidat, Toshirót és Kurát is érinti e rövid életű társaságból, Baku pedig eltűnik az akció során, és halála nagy érzelmi válságot vált ki Sanjuróban (azaz benned). A tragédia nem szakítja félbe a karrieredet, de megkeseredetté és legyőzötté tesz.

Admiral Akkaraju

Akkaraju parancsnok az UCASF Leviathanon, az UCA parancsnoki hajóján. Felesége halála mély nyomot hagyott benne, és ez megváltoztatta életszemléletét is. Az életet elutasította, és ebben az elutasításban adta át magát a feladatának, azonban a bázisa elleni

terroristatámadás e hozzáállás feladására kényszerítette. Megerősíti hírnevét, és alapos férfiú lévén becsülettel kiérdemli az admirálisi rangot.

Kathryn Akkaraju

Anyja halála után Kathryn megesküszik, hogy soha többé nem fog fegyvert embertársaira. Mindenki aggódik, amikor harcos katonaként csatlakozik az UCASF-hez, és apja csillaghajóján, a Leviathan zászlóshajón megszerzi a kommunikációs szakértői tisztséget. Kathryn fő célja a küldetés során a kapcsolattartás, jelenleg a vizsgájával foglalkozik.

Kura Akkaraju

Az Akkaraju családban mindenki katonai érvényesülés útjára lép – Kura jól megérdemelt dicséretet kapott páratlan lövésztudományáért, harcászati leleményességéért és kitartásáért. Megpróbálja kezelni a bánatát, amit anyja halála okozott, s ugyanilyen határozottsággal döntötte el, hogy kiváló katona lesz. Tartós haragja, amit apja iránt érez, meggátolja a tanulásban és elvakulttá teszi. A katonai iskola után Kura szerelmes lesz Sanjuróba, szeszélyességéből „kigyógyítja", majd gondtalan utat mutat az élet felé, és végül felülkerekedik a saját nehézségein is.

Toshiro

Mélyen érintette, hogy elveszítette a szüleit. Toshiro (Sanjuro bátyja) mogorva, zárkózott gyerek volt, aki békésebbnek találta, hogy a saját fantáziavilágában éljen, mint a többiekkel. Bár megirigyelte Sanjuro népszerűségét és nyers magabiztosságát, de soha nem vált veszélyesen féltékennyé, míg Sanjuro és Kura szerelmesek voltak.

Ryo Ishikawa

A hatalmas Shogo üzleti birodalom örököse. Házsártos és leleményes – az általa kitervelt cselszövések csak kegyetlenségével mérhetők össze.

Baku Ogata

Baku vézna volt az akadémia alatt, ezért gyakran kegyetlen tréfák célpontjául szolgált, amíg Sanjuro és Toshiro támogatni nem kezdték. Baku ugyancsak szerelmes volt Kurába gyermekéveik alatt, de soha nem osztotta meg vele igazi érzéseit.

Hank Johnson

Hank életre szóló álma, hogy UCASF-pilóta lehessen. Hibátlan, örökölt tulajdonságai révén előtérbe kerül, és kimagasló fizikuma csak fokozza sikerességét. Feleségével az elhagyott Avernus városban geotermikus monitorállomást működtet az Andra Biomechanika cég megbízásából. Keresi a lehetőséget, hogy hőssé váljon.

Mivel is irtsuk embertársainkat, avagy a mobil harci eszközök

- ***Armacham Ordog javított 7-es változat***
  Az Ordog egy a mérnökök által sokáig és magas szintre fejlesztett eszköz. Gyors és ellenálló, fényes megjelenésével, illetve visszafogott kialakításával azonban ijesztő az ellenfél számára. A jármű üzem-

Friss hírek a játékok világából

Kiadták a Legends for Linuxot

A Legends fejlesztői bejelentették a Legends) kiszolgáló alapú Linuxhoz programot. A Legends egy FPS stílusú, többjátékos program, ami a Torque grafikai motort használja (GarageGames fejlesztés; ➲ http://garagegames.com/).
➲ http://hosted.tribalwar.com/legends

Wolf: ET 2.56-os kiadás

Kiadásra került a 2.56-os változatú Wolfenstein: Enemy Territory. A fájlok a következő helyekről tölthetők le:

- et-linux-2.56.x86.run.torrent (➲ http://zerowing.idsoftware.com/BT/torrents/et-linux-2.56.x86.run.torrent) (teljes változat, 258 MB)
- et-linux-2.56-update.x86.run.torrent (➲ http://zerowing.idsoftware.com/BT/torrents/et-linux-2.56-update.x86.run.torrent) (2.55; 2.56-os folt, 6,7 MB)
- et-linux-2.56.x86.run (➲ http://3ddownloads.com/linuxgames/wolf/et/et-linux-2.56.x86.run) LG/3DD (258 MB)
- et-linux-2.56-update.x86.run (➲ http://3ddownloads.com/linuxgames/wolf/et/et-linux-2.56-update.x86.run) LG/3DD (6,7 MB)

➲ http://www.castlewolfenstein.com/

Linux Unreal Tournament 2004 kiadási hírek

Az UT2004 támogatni fogja a Linuxot (mind ügyfél-, mind kiszolgáló terén). Ugyanazon elv szerint, mint az UT2003 esetében, a linuxos támogatás benne lesz a megvásárolt UT2004 dobozában. Mire ez a cikk megjelenik, feltehetően kiadásra kerül az első előzetes is.

Neverwinter Nights-frissítés

Kiadták a Neverwinters Night Linux-ügyfél 1.32 változatát. A második ügyfél olyan javításokat tartalmaz, amelyek kiküszöbölik a felhasználók jelezte hibákat (például a játékot nem lehetett futtatni). Az ügyfél úgy lett elkészítve, hogy jobb megfelelőséget nyújtson az ismertebb videokártyáknak és -meghajtóknak.
➲ http://nwn.bioware.com/support/patch.html

Botfa Zsolttal készítettünk „elektronikus interjút" a most megjelenő hazai fejlesztésű program kapcsán.

Várhatóan e hónapban jelenik meg a Terminator 3: War of the Machines program, aminek a fejlesztését ugyanaz a Clevers cég készíti, mint a korábban elhíresült Screamer 4×4 programot (magyar fejlesztés). Lehet, hogy csak az én figyelmemet kerülte el a dolog, de eddig nem sokat halottam a játékról. Külön köszönet *Botfa Zsolt*-nak, hogy a kipróbálás időszakában – mégha csak egy-két levél erejéig is – a rendelkezésemre állt.

Kosztadinovszki Norbert: *Hogy haladtok a kipróbálással?*
Botfa Zsolt: Nos, nyakig ülünk a munkában, így csak nagyon kevés adattal tudok szolgálni... Mindenféle hivatalos tájékoztatás közzé tétele a kiadói oldal jogkörébe tartozik, ezért arra kérnélek, hogy elsődlegesen a *Kapcsolódó címeknél* felsorolt oldalakon kutakodj. Az ott szereplő anyagokat tudom egy kicsit bővíteni, illetve módosítani néhány morzsányi hírrel, tőmondatokban.

- **Karakterek**
  Human (ember): Hunter, Heavy Hunter, Supply, Scout és egy különleges karakter (ez a Terminátor-rajongóknak ismert lesz, hiszen ő Arnold).
  Terminátor: T900, T900 Supply, FK, T1.
  Nem játszható karakter: HK (ami a levegőből sorozza a humánok állásait, s csak az egyik terminátorosztály hívhatja meg).
  A játék részét képezik a Mesterséges Intelligencia által irányított karakterek is, amelyek segítenek a humán játékosoknak.
- **Járművek**
  Mindkét oldalon rendelkezésre állnak. Akadnak közöttük olyanok, amelyek nehézpáncélzattal erősítettek, és olyanok, amelyek fegyverrel felszereltek; és természetesen vannak szállítójárművek is. A játékosok a vezető, az utas és a lövész bőrébe bújhatnak.
- **Fegyverek**
  A fegyverek használata elkerülhetetlen, közel húszféle került be a játékba. Ha a fegyverarzenált nézzük, kézi lőfegyverekkel, gránátokkal, gépjárműre vagy állványra szerelt fegyverzettel találkozhatunk.
  A fegyverek egy része a mai korszerű fegyverek közül került ki, de találkozunk néhány futurisztikus változattal is (például plazmafegyverekkel). Természetesen egyes fegyverek többfunkciósak, némelyik fegyver gránátlövedékek kilövésére is képes. És meg kell említeni, hogy a játékban célkövetővel ellátott fegyverek is szerepelnek, amelyeket sikeresen használhatnak a repülő szerkezetekkel szemben.
- **Pályák**
  Több különböző stílusú pályát találhatunk: lesznek nyitottabb és zárt pályák, pályák épület belsejében, a belvárosban vagy a városon kívül. Lesznek olyan pályák, amelyek a jelenben játszódnak, míg mások a sötét jövőt tárják elénk, tehát mindenki megtalálhatja azt a fajta pályát, melyen játszani szeretne.
- **Játékmódok**
  Termination, Mission, Team Deathmatch. Lehetőség nyílik egyjátékos, helyi hálózatos többjátékos és internetes többjátékos módra is.

Amennyire lehetséges, igyekszünk valóssá tenni a harcoló felek készségeit és képességeit. Jelen esetben nem egy szokványos FPS játékról van szó, ahol a két oldalon hasonló harcértékkel bíró fél áll. Az egyik oldalon az erős, pontos, bár lassú mozgású terminátorok állnak, míg a másik oldalon a fürge, bár gyengébb emberek próbálják felvenni a harcot a gépek ellen. Ez az ellentét nagyon érdekessé teszi a játékmenetet. A terminátoroldalon szó szerint mint a gép nyomulunk előre, nem ismerve félelmet és nem is keresünk fedezéket. Az emberek oldalán pedig a pálya összes búvóhelyét kihasználjuk és tökéletes csapattaktikával győzhetjük le a gépek hadát.Tehát a játék végkifejlete azon múlik ki hogyan tudja kihasználni az általa játszott karakter képességeit.

K. N.: *Milyen grafikai motor fog futni a játék „lelkében"?*
B. Zs.: A játék egy saját fejlesztésű motort használ (Vision Graphics Engine). Egy sor hatás bizonyítja a motor képességeit, amelyek a játék minden elemén megjelennek (fegyvereken, karaktereken, járműveken stb.). A fegyvereknél látható a torkolattűz, a kirepülő hüvelyek, a fegyver mozgása, a lövedékek becsapódása. A járműveknél megfigyelhetők a lövedékbecsapódások és az ütközések nyomai, lángok, robbanások. A pálya különböző pontjain füstöt, gőzt, különböző felvillanásokat, keresőfényeket is láthatunk. Életszerű látványt nyújt a használt fény–árnyék technika. Például a torkolattűz megvilágítja a környezetet. Világosabb helyről árnyékosabb helyre történő mozgáskor jól követhető a fényváltozás.
De a grafikus megjelenés mellett meg kell említeni a hanghatásokat is, amelyek együttesen alapozzák meg a játék hangulatát.

K. N.: *A ➲ http://www.3dgamers.com/games/terminator3wm/ címen elérhető anyag szerint a játék az ATARI szárnyai alatt kerül kiadásra. Mivel az ATARI van az UT2003 és UT2004 mögött is, felmerül a kérdés, hogy a T3-nak lesz-e futtatható linuxos része, amivel a windowsos telepítő CD-ről fel lehet rakni a programot Linux alá?*
B. Zs.: Nincs tudomásom róla, hogy az ATARI tervezi a játék linuxos megjelenítését.

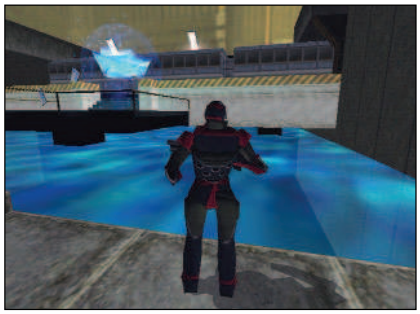
K. N.: *Az ellenfelek és játékosok felépítése hogyan történik, mennyire érvényesül a manapság mérvadónak tűnő csontvázfelépítés?*
B. Zs.: Természetesen mi is követjük az úgynevezett csontvázmodellt. Ez az animációk megvalósításához és a különböző mozgások megvalósításához elengedhetetlen.

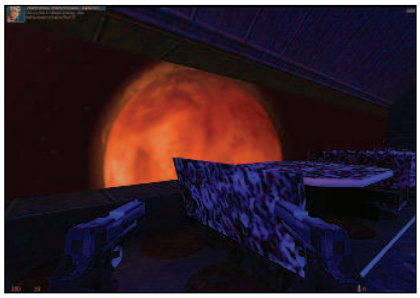
K. N.: *Megtudhatunk-e valami újabbat a játéktörténetről?*
B. Zs.: Különálló történet nem tartozik az egyjátékos módhoz. A lényeg a Terminátor-mítosz hangulatának a megteremtése.

K. N.: *Felhasználták-e és ha igen, milyen mértékben a Screamer 4×4-ben szerzett tapasztalatokat?*
B. Zs.: Az ott megteremtett fizika jó alap a mostani játékhoz. A járművek viselkedésénél nem kell szimulátorhoz hasonló viselkedés, de a járművek, a karakter, a fegyverhasználat mind a fizika törvényeinek megfelelően működik.

K. N.: *Hol tart a kipróbálás és, milyen állapotban van a játék? Kik vesznek részt a tesztelésben?*
B. Zs.: Ki sem látszunk a munkából. Amúgy a kipróbálás szervezését az ATARI (➲ http://www.betatests.net) irányítja. Remélem, ez a néhány hírmorzsa is felkeltette az olvasók érdeklődését a játék iránt.

Kosztadinovszki Norbert





módot használva gyorsabb és eléri azt a végsebességet, amit az UCA nem képes felülmúlni.

- ***UCA Enforcer Mark VII***
  Az Enforcer kemény, lekerekített MCA, ez egyensúlyozza ki érzékeny mozgását az erős páncélban. Azoknak a pilótáknak eszményi, akik szeretik a gépet, mert minden helyzetben megbízható.
- ***Shogo Akuma 12 változat***
  Az Akuma könnyű, gyors, ügyesen manőverezhető MCA, olyan helyzetekre tervezve, ahol a bonyolultság és a gyorsaság fontosabb, mint a tartósság. Tökéletes választás azok a pilóták számára, akik kedvelik a zűrzavar előli megfutamodást vagy a gyors ugrásokat.
- ***Andra 25 Ragadozó***
  Az Andra az utolsó választható MCA. Tökéletesen alkalmas harci

Rendszerkövetelmények

Processzor: Pentium 200 3D-gyorsítással, PentiumII/233 programbeli gyorsítással.

Memória: 32 MB, de 64 MB az ajánlott.

Merevlemez: 450 MB a teljes telepítéshez.

Rendszermag: 2.2.13 vagy frissebb változat.

C könyvtár: glibc 2.1.

Grafika: grafikus kártya, amely X-kiszolgáló-megfelelő, XFree86 3.3.6 vagy 4.0-s ajánlott; GTK 1.2 felhasználói felület könyvtár.

Hang: OSS- vagy ALSA-megfelelő hangkártya.

feladatokra: erős szerkezetű karral, megfélemlítő külsővel egyedülálló harci gép. Sajnos teherbírása nincs egyenes arányban az árával, a gyorsaságával és a sokoldalúságával.

- ***Fegyverek***
  – Kawamori A74 Pisztoly
  – Vollmer GA-14 CAW Vadászfegyver
  – Aegis AT-S4 Mesterlövész puska
  – Skalla kézi M25 Géppuska
  – Celsior AS-10 Impulzus karabély
  – Vollmer VK-75 Lézerágyú
  – Ludenarms MOD-4 „Bikabelező”
  – MT-101 „Eltipró”.
- ***Energiafeltöltések***
  – elsősegélycsomag
  – testvédő
  – energiaegységek
  – páncél-helyreállító egység
  – frissítések, kiegészítések.

Telepítés

Tegyük be a Shogo CD-t a CD-ROM meghajtóba, és csatoljuk fel a lemezt. Természetesen rendszergazdai (root) jogosultsággal lépjünk be a befűzött CD könyvtárba, és a következő parancsot adjuk ki:

```
./setup.sh
```

Amennyiben Gnome vagy KDE grafikus felületet használunk, egy ikon fog beköltözni a menübe.
Csak követni kell a képernyőn megjelenő utasításokat, és a 453 MB felmásolását követően felköltözik rendszerünkre a Shogo. A telepítés után a Hyperion szokásának megfelelően meg kell adnunk a sorozatszámot. Enélkül nem tudunk játszani és hiába hagyjuk ki, a `shogo` parancsra a játék nem fog futni.
A telepítő természetesen konzol alól is működik, ebben az esetben szöveges alapú telepítőt használva tudjuk a Shogót a Linuxra telepíteni. Gondolom, mindenkinek egyértelmű, hogy a futtatáshoz ebben az esetben is szükségünk van X-kiszolgálóra.
Ha valamikor úgy döntenénk, hogy le akarjuk törölni a programot, akkor a telepítőkönyvtárába lépve a `./uninstall` parancsot kell futtatnunk.

A Shogo indítása és beállításai

- ***Az indító***
  Mint már említettem, ha a Gnome vagy a KDE grafikus felületet használjuk, a telepítőprogram egy parancsikont helyez el a menübe. Vagy ezt, vagy a konzolon kiadott `shogo` parancsot futtatva juthatunk el a Shogo indítópanelhez.

Az első alkalommal, amikor használatba kívánjuk venni a Shogót, az indítóképernyő fogad bennünket – ezt ki lehet kapcsolni, ha a jövőben semmilyen beállítást nem akarunk módosítani a programon. Az indító öt beállítófüllel bír, ezeket használva tudjuk egyéni ízlésünknek vagy gépünk erőforrásainak megfelelően beállítani a program futását.

- ***Címoldal***
  Ha a Shogót a legutolsó aktív beállításokkal szeretnénk futtatni, nem kell mást tennünk, mint a ***Launch*** gombra kattintani. Ugyanitt lehet a többjátékos módot is indítani. Végül, ha semmiképpen nem akarunk játszani, nyomjuk meg a ***Cancel*** gombot. A kiválasztó négyzetbe kattintva megszabadulhatunk a többszöri panelnézegetéstől, de ha valamit sikerült elállítanunk, azt már csak kézzel, a konfigurációs fájlokban történő hosszas kereséssel lehet kijavítani.
- ***Megjelenítési beállítások***
  A lap lehetővé teszi, hogy kiválasszuk a leképezéshez használt eljárást, valamint a hozzá tartozó felbontást. A teljes képernyő lehetőség minden esetben működik, érdemes kiválasztani. A támogatott OpenGL-gyorsítással rendelkező kártyák kínálata olyan széles, hogy felsorolni sem érdemes őket. Ez annak köszönhető, hogy a játék nem mai „gyerek”. Programbeli gyorsításra csak akkor lehet szükség, ha nem rendelkezünk 3D-kártyával, de ebben az esetben is legalább 15 vagy több bites színmélységet tudó kártyára van szükség.
- ***Audiobeállítások***
  A Shogo az SDL függvénykönyvtárat használja mind a megjelenítéshez, mind a hangok leképezéséhez. Mivel a program jócskán benne van a korban, így olyan régi SDL-fájlokat keres, amiket már senki sem használ (ugyanaz a gond, mint a SiN esetében). A CD szerencsére tartalmazza

azokat a régi SDL-fájlokat, amelyek szükségesek, de egy egyszerű közvetett hivatkozás létrehozásával is orvosolható a hiányosság (lásd a SiN-ről szóló cikket).
Ugyanezen a lapon adhatjuk meg, hogy milyen mértékben történjen meg a hangcsatornák keverése. A virtuális csatornák használatával egyszerre több hang préselhető ki a hangkártyából. Természetesen minél több hangot próbálunk meg belepréselni a csatornákba, ezzel egyenes arányban nő a processzor kihasználtsága. Drasztikus esetben a hang akadozhat, a minőség viszont egyre jobb lesz. A Hyperion a következő beállításokat ajánlja:

- ***általános nyolccsatornás hanghoz Pentium II, 300 MHz-es gép***.
- ***16 csatornához K6-II/400-ast, Celeron 300-as processzort vagy még ennél is jobbat tartalmazó gép szükséges.***

Általános esetben körülbelül 20 csatorna mind a játékhoz, mind gépünk erőforrásainak a kíméléséhez elegendő. A játékban a ***Audio*** menüben lehetőség nyílik a hangerő beállítására, a csatornákat azonban futás közben már nem lehet beállítani, csak azt megelőzően.

- ***Haladó szintű beállítások***

A haladó szintű beállítások – mint neve is mutatja – csak valóban azoknak a felhasználóknak van fenntartva, akik tudják, hogy mit és miért csinálnak. Rossz vagy téves beállítások esetén a program nem fut, lelassul, illetve bizonyos dolgok, képek, hatások nem jelennek meg tökéletesen. Az alábbiakban összefoglaljuk, hogy a letiltható beállítások milyen lehetőségeket és veszélyeket is tartalmaznak. Minden esetben fontoljuk meg, hogy mit és miért kapcsolunk ki!
Sound: letiltja a digitálishang-kimenetet. Megjegyzésként szeretném megemlíteni, hogy játék során vannak olyan helyek, ahol fontos a hallott hang megértése, és természetesen halljuk, amit Hank vagy más szereplők mondanak–üzennek nekünk.
Music: letilthatjuk a CD-audiolejátszást. Hm, ennek alapvetően túl sok értelme nincs. Ha nem akarunk japán zenét hallgatni a játék közben, akkor mindenképpen előnyös.
Movies: letilthatóak a kezdő és a logófilmek a játék indításakor. Ebben az esetben a főmenü hamarabb fog megjelenni.
Light Mapping: letiltható a fény-hozzárendelés. A fény-hozzárendelés egy olyan folyamat, amelynek során egy sokszögön egy második textúrát alkalmaznak, ami olyan hatást ad a fénynek vagy az árnyéknak, mintha valós, vetett fény jelenne meg a felületen. A fény-hozzárendelés nagymértékben feljavítja a kimenő kép minőségét, ezért a minőségért azonban némi sebességgel kell fizetnünk. A fény-hozzárendelést letiltva a program a ***vertex lighting*** használatával megpróbálja szimulálni a hatást, de a minőség nem lesz olyan jó, viszont a sebesség javulni fog.
Csak megjegyzés, hogy az eredeti leírás szerint néhány grafikus kártya alapértelmezettként letiltja a szóban forgó szolgáltatást, mivel nem tudja kezelni. Ez alapvetően a Permedia2 alapú kártyákra érvényes. Manapság ugyanakkor senki nem használ ilyet.
Fog: letiltható a köd. A ködtechnika szimulálja a környezet mélységét, és atmoszférikus hatásokkal kevert pixelszínekkel látja el a ködöt. A köd színe függ attól, hogy milyen távol vagyunk tőle. A ködhatásnak a `FOG_COORD_EXT` támogatásra van szüksége, amit az OpenGL könyvtárnak kell nyújtani. Ha ez nem érhető el vagy nem támogatott, akkor önműködően letiltásra kerül. Itt is letiltható ez a hatás.
Line Systems: letiltható a vonalrendszer. Néhány hatás – például torkolattűz esetében – több vonallal kerül megrajzolásra, ami olyan benyomást kelt, mintha röntgensugarakkal rajzolnánk. Ez a kiválasztódoboz letiltja a vonalrendszert, amire csak akkor van szükség, ha az OpenGL könyvtár hibás.
Model FullBrights: letiltható a teljes ragyogáshatás a modelleken. A teljes ragyogás módszere szimulálja az izzó objektumokat, például a plazmafegyver csövét. Ez a lehetőség a képleképezésnél időcsökkenéssel jár – ha letiltjuk, szintén sebességnövekedést kapunk. Számos izzó, fényes modell válik láthatóvá a letiltást követően, ha a hatás rosszul működik és roszszul rajzolódnak ki a tárgyak. Általában nincs szükség a modellek ilyen jellegű hatásának a letiltására.

A következőkben az „engedélyezhető” lehetőségeket foglalja össze, ezek olyan beállítások, amelyek pluszlehetőségekkel látják el a program futását. Ezek egyes rendszereken esetleg galibát is okozhatnak.
Polygap Fixing: hogyha a Polygapfixálás engedélyezve van, a játék a réseket a sokszögeken megpróbálja hálóvá alakítani, hogy eltűnjenek a sokszögek csúcsain keletkező pontatlanságok. Ez a folyamat lelassítja a leképezést. Csak akkor kell bekap-

KAPCSOLÓDÓ CÍMEK

- http://www.t3war.com
- http://www.atari.com
- http://www.gamespy.com/games/5999.shtml
- http://gamespy.com/interviews/june03/t3wotm/
- http://www.gamespy.com/e32003/preview/pc/1002504/
- http://www.gamespydaily.com/news/fullstory.asp?id=5185
- http://us.atari.com/press/print.php?id=370

csolni ezt a lehetőséget, ha a sokszögek találkozásánál látszanak a lyukak.
Pixel Doubling for Movies: a filmek csak 320×240 pixeles méretűek, így nagyobb felbontású képernyőn rosszul láthatók és túl kicsik. Ezen a gondon segít a beállítás, ami a filmeket az eredeti gyári méretről mindkét irányban teljes, 640×480 felbontású filmmé feszíti ki.

- ***Konzolgomb***
  Fontos megemlíteni, hogy a konzolgomb több célból is fontos. Amennyiben egyéni vagy letöltött ***MOD*** lehetőségeket akarunk használni, azt a konzolról is megtehetjük. A ***Choose*** gombra kattintva megadható, hogy milyen gombbal szeretnénk meghívni a játékon belül a konzolt.
  A konzol használatával lehet begépelni a cheat-kódokat is, ezeket a CD-mellékleten is megtalálhatjuk.
- ***A testreszabó oldal***
  Ez az oldal lehetővé teszi, hogy egyedi, hozzáadott modulokat töltsünk be a játékba. Ezeket a fájlokat nevezzük egyszerűen REZ fájloknak. A REZ fájlok több hozzáadott értéket, adatot is tartalmazhatnak, és a típusuk is eltérő lehet. Lehetnek bővítőadatok a játékhoz, ilyenek az új többjátékos vagy egyjátékos szintek, esetleg újabb küldetések, játékjavítások és további nem hivatalos bővítmények. A kezelés nagyon egyszerű: a bal oldali ablak az elérhető, feltelepített bővítményeket tartalmazza, míg (ha az ***Add*** gombot használjuk) a jobb oldalon a játék indításakor betöltendő bővítmények szerepelnek. Mindig ellenőrizni kell, hogy a jelölő négyzetbe ki van-e pipálva, mivel csak akkor töltődnek be a bővítő REZ fájlok a játékba, ha igen.
  Fontos, a REZ fájlok egyéni alkönyvtárakban találhatók, vagyis ide telepíthetők fel rendszergazdai jogosultság birtokában. Minden esetben, amikor ilyen REZ fájlokat használunk, győződjünk meg az eredetéről, mivel ezek a fájlok futtatható állományok, s előfordulhat, hogy a betörések melegágyává tesszük a gépünket. A Shogót a saját felhasználói könyvtárunkból futtassuk, így mérsékelhető a betörés kockázata.

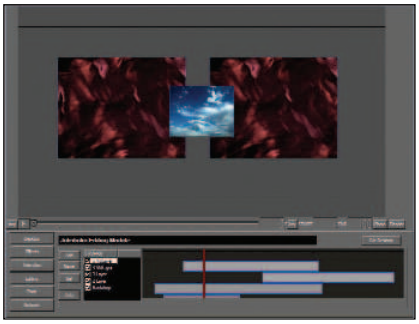
A CD-mellékleten megtalálható egy parancsállomány, ami a Shogó indító parancsfájl javított változata. Miről is van szó? A parancsállomány az eredeti ***shogo*** fájlon alapul, de kijavítja annak hibáit. A gond az, hogy a feltelepített ***shogo*** parancsállomány `devfs` használatával akarja kezelni és olvasni a CD tartalmát. Mihelyt ez nem sikerül neki, értelmetlen üzenetet ad vissza: „Tegye be a Shogo CD-ROM-ot!” – holott benn van és be is lett fűzve.
Akit bővebben érdekel a téma, az olvassa el a ***/usr/src/linux/Documentation/cdrom/**** és a ***/usr/include/linux/cdrom.h***, hogy megértse a ***subchannel*** és a ***toc*** kezelését. A CD-mellékleten megtalálható a Shogo-demó, a SiN-demó, valamint a CD-ken lévő mpg-fájlok. A demók ki vannak próbálva, és futtatásuk után ugyanabba a könyvtárba csomagolják ki a tartalmukat, ahonnan a játékot is indíthatjuk. Így a Hyperion jelenlegi teljes linuxos játékkínálata dokumentálva és a demók közre lettek adva.

Kosztadinovszki Norbert
(kosztadinovszki@diaelec.hu)
Linux- és játékmániás számítógépőrült.

Hasznos apróságok

Teljes fejlesztéskezelési rendszer: Gforge

Megjelent a GForge (➲ http://gforge.org) webes együttműködési programfejlesztési rendszer 3.0-s változata. A Gforge szolgáltatásai közé tartozik a hibakövetési rendszer, a CVS változatkezelés, a webes CVS-felület, illetve az archivált



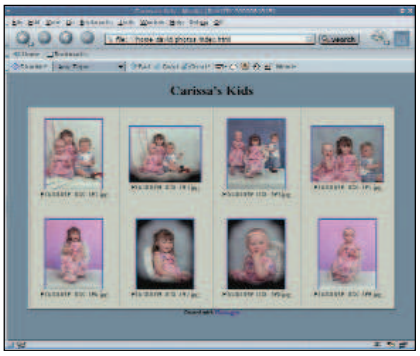
Mailman levelezési listák. Ezenkívül a 3.0-s változat egy új projektkezelési rendszert tartalmaz, amely Gantt diagramok készítésére, nemzetköziesítésre alkalmas, és egyszerűen telepíthető. A GForge a SourceForge.net működését biztosító program rendszeresen karbantartott, szabad forrású fejlesztőeszköze. Futtatásához PostgreSQL, Apache, OpenSSL és PHP szükséges.
Don Marti

iBackup

Ez a mentési segédprogram nem a rendszerről készít biztonsági másolatot, hanem a beállításfájlokról. A mentendő adatokat egy HTML-fájlban tárolja, amit a tarprogrammal becsomagol, illetve igény szerint tömörít is. Az iBackup azt is lehetővé teszi, hogy ezt a fájlt egy másik rendszerre megőrzés céljából feltöltsd. Így könnyen és gyorsan elvégezhető a fontos adatok (például a jelszó, a csoport, az árnyékfájlok, BIND-fájlok stb.) mentése és visszaállítása. Futtatásához BASH, `tar`, `gzip`, `ifconfig`, `netstat`, szabványos Unix-eszközök szükségesek.
➲ http://www.linuks.mine.nu/ibackup
David A. Bandel

PhotoGen

A PhotoGen is remek, könnyen használható program, amellyel webes fotóalbumot készíthetsz minden nehézség nélkül. Az ilyen programok segítségével bárki létrehozhat egy miniatűröket tartalmazó indexoldalt, és a teljes könyvtárat közvetlenül kiteheti egy webkiszolgálóra. Erre a könyvtárra



hivatkozva azután mindenki megtekintheti a fényképeket.
➲ http://shawley.myip.org/projects/photogen.php
David A. Bandel

jigl

Ez a Perl-parancsfájl túlságosan is egyszerű. Létrehozol egy könyvtárat, ebbe bemásolod az összes olyan fájlt, amit fel akarsz tenni egy weboldalra, végül a könyvtárból lefuttatod a ***jigl.pl*** parancsfájlt. A program azonnal elkészíti a miniatűröket, az azokhoz tartozó weblapokat, és egyesével végignézheted őket. Különböző, könnyen kezelhető sablonok segítik a testreszabást. Előfeltételei: Perl, ImageMagick, jhead.
➲ http://xome.net/projects/jigl/
David A. Bandel

Inventory

Ez a program mindenre képes. Rendezheted, kereshetsz benne, kiteheted, behozhatod. Osztályokat és alosztályokat hozhatsz létre, az oszlopok típusa lehet logikai, egész vagy karakterlánc. Valójában ez a csinos kis segédprogram bármilyen alkalmazásként használható, nemcsak leltárkészítésre, és szigorúan magánügy, hogy ki mit kezd vele. Kétlem, hogy bárki ellenezné, ha például feladatlistaként vagy címjegyzék gyanánt használnád. A papíron és a ceruzán kívül kevés ilyen sokoldalú dolog van, olyan pedig talán nincs is, ami ennyire hatékony lenne. Futtatásához szükséges: libgtk-x11, libgdk-x11, libatk, libgdk_pixbuf, libm, libpangoxft, libpangox, libpango, libgobject, libgmodule, libdl, libglib, libmysqlclient, glibc, libXi, libXext, libXft, libX11, libz, libXrender, libfontconfig, libfreetype, libcrypt, libnsl, libexpat.
qballsinventory.sourceforge.net
David A. Bandel

sudo

Eme kis program segítségével kitüntetett felhasználóként olyan engedélyezett parancsokat futtathatunk más felhasználók nevében, amelyet jogosultsági szintünk egyébként nem tenne lehetővé. Ez olyankor válhat hasznossá, ha például csak a rendszergazda menthet egy adatbázist, ám azt szeretnénk, hogy az egyik munkatárs is naponta meg tudná tenni anélkül, hogy rendszergazdai jogosultságokkal kellene rendelkeznie. Ilyenkor csak engedélyezni kell számára az adott parancsnak sudóval rendszergazdaként történő futtatását!
Komáromi Zoltán

Linux Journal 2003. 112, 114. számok

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

Don Marti
(dmarti@linuxjournal.com)
A Linux Journal szakmai szerkesztője.

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